UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

A Chatbot for Automatic Question Answering in the Information Technology Domain

João Miguel Vila Antunes

Mestrado em Engenharia Informática
Especialização em Interação e Conhecimento

Dissertação orientada por:
Prof. Doutor António Manuel Horta Branco
e Doutor Luis Manuel dos Santos Gomes

2019

Agradecimentos

Esta dissertação de mestrado só foi possível graças ao apoio recebido ao longo desta minha etapa. Correndo o risco de injustamente não mencionar alguém, deixo aqui os mais sinceros agradecimentos às pessoas que, direta ou indiretamente, contribuíram para este trabalho:

- Ao Professor Doutor António Branco e ao Professor Doutor Luís Gomes, pela orientação, total disponibilidade e apoio prestado;
- Ao Portugal 2020 pelo financiamento no âmbito do projecto ASSET;
- Aos pais João Antunes e Paula Antunes, por me terem permitido fazer este curso e por fazerem tudo para que nada me falte;
- Aos avós Ana Morais, Alberto Vila, Lúcia Correia e José Antunes e a todos os restantes membros da família, por me terem acompanhado desde que nasci;
- À namorada Mafalda Sequeira, pela sua paciência e cumplicidade;
- Aos amigos que conheci ao longo do meu percurso académico Bernardo Reis, Bruno Pereira, Carlos Brito, Dharmite Prabhudas, Filipe Pereira, Iúri Matos, João Bastos, João Batista, João Cardoso, João Miguel Rodrigues e Sara Nascimento, por terem partilhado comigo tanto os momentos positivos como os menos positivos;
- Ao amigos de Sintra André Faria, Beatriz Louro, Nuno Bartolomeu, Ricardo Silva e Tiago Costa, pelas piadas secas e gargalhadas partilhadas;
- A toda a família Lisboa Navigators, sobretudo àqueles que lutaram ao meu lado, por me permitirem praticar um desporto exigente que requereu a assimilação de novos valores que certamente me irão continuar a acompanhar diariamente;
- Às restantes pessoas que conviveram diariamente comigo no NLX Andrea Teixeira, Frederico Apolónia, João António, João Silva, Małgorzata Salawa, Márcia Bolrinha, Rodrigo Santos, Rúben Branco, Sara Grilo e Tânia Oliveira pelo ótimo ambiente de trabalho proporcionado.

Resumo

Os *chatbots* têm sido alvo de grande estudo por parte da comunidade de Inteligência Artificial e de Processamento de Linguagem Natural e o seu futuro parece promissor. A ideia de automatizar conversas através do uso da tecnologia é bastante interessante para muitas empresas visto que fornece vários benefícios a um preço relativamente baixo. Por exemplo, pode dar apoio ao cliente a tempo inteiro, visto que consegue comunicar com um grande número de pessoas ao mesmo tempo e pode ser utilizado como ferramenta de automatização de trabalho repetitivo. No entanto, os sistemas atuais não conseguem por vezes acompanhar as expectativas cada vez mais exigentes dos utilizadores e por vezes falham em oferecer uma experiência tão simples e eficiente como gostaríamos.

A falta de conjuntos de dados para treinar modelos é um dos principais problemas enfrentados pelos investigadores visto que para um conjunto de dados ser útil, precisa de ter um número de conversações muito elevado. Outro problema frequente relaciona-se com a dificuldade em desenvolver chatbots capazes de criar diálogos convincentes, semelhantes aos que seriam feitos por humanos num determinado contexto.

Esta dissertação tem por objectivo a construção de um *chatbot* que possa ser usado para responder a questões num ambiente de apoio informático. Este sistema deverá ser capaz de analisar uma determinada questão e, com base na informação com que foi treinado, devolver uma ou um conjunto de respostas correctas possíveis. Face ao atual interesse da comunidade de NLP relativo aos *chatbots* generativos, que dado um contexto, geram, usualmente palavra a palavra, as próprias respostas e devido à maior facilidade em adaptarem-se a novas perguntas que não existem no conjunto de dados de treino, este tipo de *chatbots* foi escolhido como foco central do trabalho desenvolvido.

Assim sendo, três *chatbots* generativos foram replicados ao terem sido treinados e avaliados. Estes *chatbots* têm como nome *Hierarchical Recurrent Encoder-Decoder* (HRED), *Variational Hierarchical Recurrent Encoder Decoder* (VHRED) e *Variational Hierarchical Conversation RNNs* (VHCR). De entre estes três modelos, o HRED é o mais simples, sendo bastante semelhante a um modelo *Encoder-Decoder* básico. Para além do *Encoder* e do *Decoder*, o modelo HRED utiliza uma Rede Neuronal Recorrente (RNR) adicional que mantém informações relacionadas com o contexto da conversa atual e que é utilizada para condicionar o *output* gerado pelo modelo. Por seu turno, tanto o modelo

VHRED como o modelo VHCR são *Variational Autoencoders*, um tipo de sistemas generativos que nos últimos anos tem sido muito estudado por ter uma grande capacidade de gerar novos dados. Embora inicialmente aplicado em geração de imagens, este tipo de modelos foi aplicado ao contexto de Processamento de Linguagem Natural através do modelo VHRED, com o objectivo de ser um modelo capaz de gerar frases mais diversificadas do que os restantes modelos existentes e de conseguir capturar a informação global do conjunto de dados de treino. O VHCR, por sua vez, assume-se como uma extensão do modelo VHRED ao ter uma estrutura muito idêntica a este e foi proposto para mitigar um problema que o VHRED tem relacionado com um défice na utilização correcta de uma variável latente fulcral à obtenção dos resultados pretendidos.

Outra contribuição desta dissertação é a criação de uma ferramenta que permita extrair novos conjuntos de dados constituídos por diálogos entre humanos. Esta ferramenta utiliza o *website* Reddit, onde diariamente milhares de utilizadores partilham conteúdo em forma de perguntas, artigos e links, como fonte dos diálogos extraídos. Mais concretamente, esta ferramenta cria diálogos através da interação com bases de dados que são disponibilizadas *online* mensalmente. Um aspeto interessante desta ferramenta é o facto de permitir que conjuntos de dados de uma grande variedade de domínios sejam extraídos, o que permitirá obter conjuntos de dados relacionados com domínios nos quais ainda não exista nenhum.

Um dos conjuntos de dados que foram extraídos recorrendo a esta ferramenta é genericamente focado no domínio da informática e expresso na língua inglesa, abordando vários temas distintos, entre os quais o Ubuntu e questões relativas a linguagens de programação. Este conjunto de dados tem como nome *askIT* e foi utilizado para treinar os modelos anteriormente referidos. Outro conjunto de dados extraído é maioritariamente composto por diálogos expressos na língua portuguesa e tem como nome *Portuguese*. Ao contrário do conjunto de dados anterior, este não se foca em nenhum assunto particular, e inclui diálogos sobre assuntos muito variados entre os quais cultura e atualidades.

Para além destes conjuntos de dados extraídos por mim, o *Ubuntu Dialogue Corpus* foi também utilizado para treinar os modelos. Este conjunto de dados é uma coleção de diálogos relativos a apoio técnico sobre o Ubuntu e foi escolhido por ser de grandes dimensões e por abordar um tema semelhante ao pretendido para o meu *chatbot*.

Para avaliar os modelos referidos, duas formas de avaliação distintas foram utilizadas: uma baseada em representação semântica vetorial (*embeddings*), extrínseca ao modelo, e outra baseada na perplexidade de palavras, intrínseca ao modelo. A avaliação de modelos baseada em *embeddings*, tal como o nome indica, foca-se na análise e comparação de *embeddings*, que são vetores representativos de uma palavra ou frase. Cada *embedding* tenta capturar o significado da frase ou palavra correspondente, o que leva a que frases ou palavras semelhantes sejam representadas por *embeddings* semelhantes. Assim sendo, nesta avaliação, os *embeddings* das frases geradas pelos modelos são comparados com os

embeddings das frases de referência existentes no conjunto de dados de teste. Por sua vez, a avaliação baseada na perplexidade mede a surpresa que o modelo treinado tem a prever o conjunto de dados de teste.

Para além destes dois tipos de avaliação e através de um programa criado por mim para interagir diretamente com o modelo, foram feitas algumas questões ao VHCR treinado tanto com o *Ubuntu Dialogue Corpus* como com o *askIT* com o fim de fazer um juízo sobre as suas capacidades.

Através destas avaliações, foi possível observar que os modelos treinados com o *Ubuntu Dialogue Corpus* obtiveram melhores resultados na avaliação extrínseca, enquanto que os treinados com o *askIT* obtiveram melhores resultados na avaliação intrínseca. No entanto, com as respostas dadas pelo VHCR, concluiu-se que para o objectivo de construir um chatbot capaz de responder a questões num ambiente de apoio informático, os modelos treinados com o *Ubuntu Dialogue Corpus* seriam os mais adequados. Concluiu-se também que para este tipo de fins, os *chatbots* baseados em recuperação de informação continuam a ser os mais adequados porque as respostas geradas pelos *chatbots* generativos não parecem estar ainda num nível competitivo.

Palavras-chave: Chatbots Generativos, Interfaces Conversacionais, Conjunto de Dados, Inteligência Artificial, Processamento de Linguagem Natural

Abstract

Recently, chatbots have been thoroughly studied by the Artificial Intelligence and the Natural Language Processing communities and the future of this technology appears promising. The idea of automating and scaling one-to-one conversations using technology appeals to companies since it can provide benefits in a cost effective way. However, current systems sometimes cannot keep up with the increasingly demanding user expectations as they sometimes fail to deliver experiences that are as seamless and efficient as we envisioned them to be.

The lack of datasets to train models is one of the biggest problems faced by researchers since, for a dataset to be useful, it needs to have a very large number of conversations. Another problem researchers commonly face relates to the difficulty of developing a chatbot that is capable of generating convincing dialogues, similar to what a human would say in a given context.

For this dissertation, I developed a tool that is capable of extracting new datasets containing dialogues between humans. This tool uses Reddit, a website in which thousands of users share content daily, as source and allows the creation of datasets related to a wide number of domains.

Additionally, three state-of-the-art dialogue models were replicated and trained on two datasets of the information technology domain, one of which was extracted by the abovementioned tool. Two types of evaluation were conducted, one intrinsic to the models and the other extrinsic, and the results obtained are in line with the results reported by their original authors.

Keywords: Generative Chatbots, Conversational Interfaces, Datasets, Artificial Intelligence, Natural Language Processing

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter establishes a groundwork for the rest of the dissertation. Chatbots are introduced here and their evolution is overviewed. Then, the different types of chatbots are presented and a contextualization between them and current businesses is made.

1.1 Motivation

The idea of being able to hold a conversation with robots has fascinated people for a long time and up until a few years ago, such idea was nothing short of a work of fiction. However, people are now able to interact with devices like smart phones by dictating or writing requests to them. By doing so, it is now possible to use applications in a more intuitive way.

A chatbot is a computer program that is able to conduct a conversation via auditory or textual methods. Such technology can be used in a wide number of purposes and by resorting to chatbots, companies can now engage with their customers in a simpler way, cut down on their workload and proactively prevent minor customer service issues from becoming major problems.

As indicated below in this chapter, there are multiple types of chatbots but the ones that seem to be the current focus of the NLP community are the generative-based chatbots. As such, this dissertation aims to experiment with such chatbots in the IT domain in order to get acquainted with what these models are currently capable of.

1.2 Research Context and Goals of the Dissertation

The work developed and presented in this document was undertaken during my internship at NLX-Natural Language and Speech Group,[1] a research group for Natural Language Processing from Faculdade de Ciências da Universidade de Lisboa.

[1] http://nlxgroup.di.fc.ul.pt/

This dissertation was performed within the scope of the ASSET (Intelligent Assistance for Everyone Everywhere) project, which aims to improve automatic assistance quality on various languages for the Information Technology domain, under the grant ANI/3279/2016.

The main goal of this dissertation is to build a chatbot that can answer questions related to IT, while also having the ability of being trained to answer questions in other domains, as long as it is trained with adequate datasets. This IT chatbot may be used on a customer support level to minimize repetitive work. Even if the output returned by the system is not good enough to be shown directly to the customers, it may be used to help professionals handle customer requests.

Other than the system, a new dataset extraction tool had to be created as it can be an important resource for researchers to obtain new datasets and consequentially train models for new domains.

1.3 Chatbots and Dialogue Systems

Like their name implies, chatbots are a type of dialogue systems that can chat with the user, i.e. chatbots are systems that receive as input a recording of a spoken utterance or a written one, process it and give a response as output. Because of their adaptability to handle various tasks, chatbots can be used in a large number of contexts, as described below in Section 1.6. Moreover, their capabilities have been improving substantially over the years as we will discuss in this dissertation.

Other than chatbots, we also meet other types of dialogue systems daily: when calling to a telecommunications service provider, we might first hear the auto-informer system; when reserving airline tickets, we might interact with one that matches our needs to the available options; or when using a GPS app on our smart phone, we listen to the directions that are given. These systems also interact with the user but in a different manner than chatbots.

Next, we will go over a few chatbots that were historically relevant, from the beginning of these systems to the ones that are commonly used today.

1.4 Historical Notes on Chatbots

Chatbots have been gaining a lot of research interest, have caught public's attention and its history is already quite long as its notion has been around since the 1950s. It all began when Alan Turing theorized that a truly intelligent machine would be indistinguishable from a human during a text-only conversation. The test known as the Turing test, a method of inquiry for determining whether or not a computer is capable of thinking like a human being, was proposed by Alan Turing [1]. In the Turing test, three physically separated

terminals are used, in which two are operated by humans and the other by a computer. One of the humans assume the role of the questioner and interrogates the respondents, i.e. the other two terminals. After a particular amount of time or number of questions, the questioner is asked to decide which respondent is the human and which is the computer.

1.4.1 Some First Chatbots

Text-based dialogue systems for question answering and chatbots that simulated casual conversation started to be developed in the 1960s.

ELIZA [2] was created at the MIT Artificial Intelligence Laboratory by Joseph Weizenbaum. It simulated human conversation by matching user prompts to scripted responses giving the illusion of understanding the conversation.

PARRY [3] was implemented by psychiatrist Kenneth Colby at Stanford. This chatbot simulated a person with paranoid schizophrenia. It embodied a conversational strategy and was much more serious than ELIZA. In October 1972, in a demonstration at the International Conference on Computer Communications (ICCC), Vint Cerf, a computer science pioneer, set up a conversation between ELIZA and PARRY using ARPANET (an early packet-switching network).

Jabberwacky [4] started to be developed by Rollo Carpenter in 1981. It was designed to "Simulate natural human chat in an interesting, entertaining and humorous manner". It stored everything ever said to it and found the most appropriate thing to say based on contextual pattern matching techniques.

Dr.Sbaitso [5] was an Artificial Intelligence (AI) speech synthesis program released in 1991 by Creative Labs for MS DOS-based personal computers. It would interact with the user as if it was a psychologist, but most of its responses were along the lines of "Why do you feel that way?" rather than any sort of complicated interaction. It was designed to showcase a digitized voice.

A.L.I.C.E. [6] was introduced in 1995 by Richard Wallace. It engages in a conversation with a human by applying some heuristic pattern matching rules to the human's input but it was not able to pass the Turing test. It was the inspiration for Spike Jonze's 2013 academy award-winning film *Her*.

1.4.2 Dialogue Systems and the Internet

In 2001, **SmarterChild** [7] was released and it was available on AOL Instant Messenger and Windows Live Messenger (previously MSN Messenger) networks. It was the first chatbot that was able to pull and return info from the internet when requested. As such, it is considered a precursor of present day systems like Apple's Siri.

IBM's Watson [8] was released in 2006 and was originally designed to compete on "Jeopardy!" (a popular American general knowledge quiz TV show), having beaten two

of the show's former champions. Watson has since gone on to bigger and better things such as offering support to hospitals and health organizations.

In 2010, **Siri** [9] was released and it has since been used by Apple users, having become one of the most used "Intelligent Assistants" ever. It uses voice queries and a natural-language user interface to answer questions, make recommendations, and perform actions by delegating requests to a set of Internet services. The software adapts to the user's individual language searches, usages and preferences and the returned results are individualized. This assistant is considered a milestone for all later AI Bots and Personal Assistants.

In 2012, **Google Now** [10], which was developed by Google for their mobile search app, was released and its features are similar to Siri's: it can answer questions, make recommendations and perform actions by passing on requests to a set of web services.

Alexa [11] was released in 2015 for the Amazon Echo Device. Alexa is an intelligent personal assistant that is capable of voice interaction using Natural Language Processing (NLP) algorithms to receive, recognize and respond to voice commands.

In 2016, **Tay** [12], a chatbot created by Microsoft, caused a lot of controversy. The bot made replies based on its interactions with people on Twitter and only 16 hours after being launched, it had to be shut down due to the fact that it started to post offensive tweets and displaying an increasingly paranoid behavior.

1.5 Importance of the Topic

As chatbots are of great importance to the NLP and AI area, some competitions and challenges have been proposed related to them. Among these, I highlight the Loebner Prize competition and the Dialog System Technology Challenge given their popularity and relevance to the specific topic of this dissertation.

1.5.1 The Loebner Prize

The Loebner Prize is an annual competition in AI that was launched in 1990, it is run by The Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB)[2] and is usually held in the UK.

Computer programs are tested in the format of a standard Turing test. In each round, a human judge simultaneously holds textual conversations with both a program and a human being, using a computer. Based on the responses obtained, the judge has to decide which is which. To make things more challenging, no internet access is allowed which avoids the possibility of cheating by having a human operator pretending to be a chatbot and typing their response, as well as accessing it to obtain more information that might be useful in a particular conversation.

[2] aisb.org.uk

The winner gets a prize money of about $\$3,000$ and in addition, $\$25,000$ may eventually be given to the program that at least half of the judges cannot distinguish from a real person. $\$100,000$ is the prize that may be awarded to the program that the judges cannot distinguish from a real human in a Turing test that includes deciphering and understanding text, visual, and auditory input. This competition will end once this is achieved.

A.L.I.C.E. was the winner of this competition in 2000, 2001 and 2004 and Jabberwacky was the winner in 2005 and 2006.

1.5.2 Dialog System Technology Challenge

The Dialog System Technology Challenge (DSTC)[3] is an on-going series of research community challenge tasks. Since 2013, a number of challenges related to dialogue systems have been proposed yearly. Throughout these years, some datasets have been shared, and these are briefly described in Section 4.2.

In 2019, one of the proposed tracks, "End-to-End Conversation Modeling: Moving beyond chit-chat - Sentence Generation", was based on sentence generation, which shows how relevant the topic of this dissertation is. The goal of this task was to "move beyond chit-chat" by allowing data-driven models to access an external knowledge base. With this ability, it was expected that the models would produce more meaningful and contentful sentences. Similar to the problem addressed in this dissertation, the goal of these models was not to be completely goal-oriented but instead they should be able to carry a relevant conversation with the user.

1.6 Chatbots and Industry

The nearly simultaneous push by the likes of Amazon, Apple, Facebook, Google and Microsoft on chatbots helped to raise the popularity of this technology, and with that came, among others, companies like KiK, Slack and WeChat that helped consolidate its position on the market. Advances made by the AI and the NLP community helped to raise chatbots to a whole new level but it is clear that there is still a lot of progress to be made.

Most companies started hearing about this technology only around 2015 or 2016 and many still have to take the plunge to use it. This hesitation may stem from the fact that, according to a study conducted by Mindbowser in late 2016 [13], 75% of the businesses believe that chatbots have not yet proven themselves completely.

However, most people who did decide to use chatbots are happy with the way they are helping their business, since they can offer a lot of benefits in a cost effective way. For instance, they are able to automate repetitive and lengthy tasks so their staff can focus on more important work.

[3] http://workshop.colips.org/dstc7/

Some graphs shown in Mindbowser's study are presented in Figures 1.1 and 1.2. The Y axis in Figure 1.1 represents the percentage of people that believes chatbots will be useful on the field specified in the corresponding X axis and in Figure 1.2, it represents the percentage of people that believes chatbots will be useful for the business functions specified in the corresponding X axis. This information was obtained by interviewing more than 300 individuals from a wide range of fields including online retail and hospitality.

As we can see from Figure 1.1, there are multiple industries in which chatbots could be used. E-commerce, Insurance and Healthcare are the industries that appear to have a greater number of people believing that chatbots can be helpful in their respective industry. This may be because these industries are mostly associated with customer service, which is by far the business function that is expected to benefit the most from this technology, as seen in Figure 1.2. However, other industries and functions also present opportunities that can be explored.

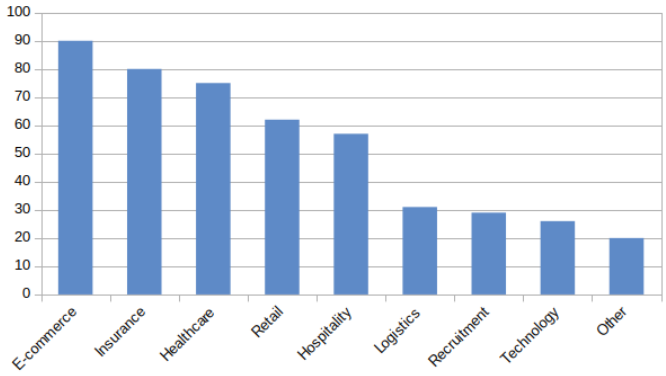

Figure 1.1: Industries that will benefit the most from Chatbots
(reproduced from 2016 Mindbowser study [13])

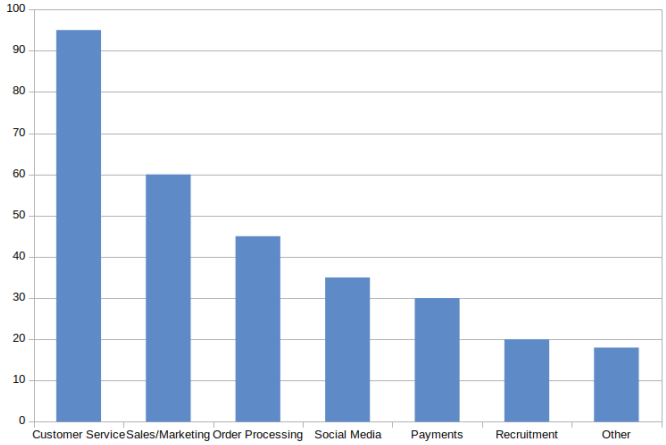

Figure 1.2: Business functions which will benefit the most from Chatbots
(reproduced from 2016 Mindbowser study [13])

1.7 Generative and Retrieval Chatbots

Machine Learning-based chatbots can either be retrieval-based or generative.

Retrieval-based chatbots are systems that try to find the best response for a given context by calculating a matching score with the context and each of the candidate responses from a list of human-written responses. The candidate response with the highest matching score is used as the final response, usually unedited, and these candidate responses are present in a predefined repository. This is the most common type of chatbots used as these are easier to develop and the quality of the output given is more easily anticipated.

As their name indicates, in contrast, generative chatbots generate, usually word by word, responses given a context. This will lead to the ability of generating sentences that are not present in the training set. This type of system is considered more flexible than the retrieval-based one, but due to the difficulty of anticipating responses and to the fact that the generated responses may not be grammatically correct, it is not so commonly used.

For the present dissertation, a generative dialogue system was chosen to be replicated. The main reason supporting this decision was because I wanted to gain an informed insight on how well these systems are currently able to learn how to model a language, and to retain useful information. Another reason was because the ability of generating sentences that are not present in the training set should make this type of system better at handling unforeseen situations. Even though this type of system may not have been as explored as the retrieval-based ones, recent studies have made drastic improvements in this kind of chatbots. Some of the models that were proposed in these studies are introduced in the following chapters.

1.8 Open and Closed Domain Chatbots

One of the most determining aspects of a chatbot is its domain, which may be qualified either as closed or open. Closed domain chatbots specialize in a particular topic and are only expected to be able to answer to questions related to it. On the other hand, open domain chatbots can have conversations about any topic, which comes with the price of generally giving lower quality responses than the ones a closed domain system would output. However, an open domain chatbot is expected to perform better on all other domains.

In general, a closed domain chatbot is expected to output higher quality responses than the ones output by an open domain one, if both systems are trained on top of datasets with similar sizes and quality.

1.9 Structure of the Remainder of this Document

This document is formed by five more chapters.

In Chapter 2, *Related Work*, some technical mechanisms and models will be discussed. Some of these mechanisms and models can be seen as precedents to the ones explained in Chapter 4, while others correspond to approaches that while briefly described in this dissertation, were not any further used.

In Chapter 3, *Reproduced State-of-the-art Models*, the models that were replicated for this dissertation are explained. A program that was developed for this dissertation to use these models as chatbots is also presented.

In Chapter 4, *Datasets and Tools*, some of the most commonly used datasets for dialogue systems are presented. This chapter also proposes a novel tool I developed that is able to extract new datasets from real conversations between humans. Two new datasets extracted with this tool are also presented in this chapter.

In Chapter 5, *Performance Evaluation*, the results obtained with different evaluation metrics in different models and corpora are presented. In addition to these results, their analysis and samples of these trained models are also presented.

In Chapter 6, *Conclusion*, this dissertation is concluded with some final notes that include a reflection of the whole dissertation and possible lines of future work.

Chapter 2

Related Work

In this chapter, models and mechanisms that are important to NLP, in general, and chatbots, in particular, are introduced. I start by introducing different types of neural networks that are used in most state-of-the-art chatbot architectures. Afterwards, some recently proposed chatbot models are overviewed to better grasp the common current model's architecture. Finally, a central model to understand the ones presented in Chapter 3 is described.

2.1 Introduction to Sequence Models

Sequence models are models that, when given a sequence as input, look at each element of that sequence and try to predict a sequence to return as output. In NLP, this sequence can be of multiple types, including a translation or a reply to a given language input.

Sequence models are typically Recurrent Neural Networks (RNNs) and several notable RNN models have been proposed over time. For this dissertation and due to their importance and relevance to the models replicated in this dissertation, I highlight the Long Short-Term Memory Networks (LSTM) [14], the Gated Recurrent Units Networks (GRUs) [15] and the Bidirectional Recurrent Neural Networks (BRNNs) [16].

2.1.1 Recurrent Neural Networks

A RNN is a network with loops that allow information to be persisted and its output is influenced not only by the current input but also by the history of inputs, something that previous neural networks did not do.

On the left side of the Figure 2.1, a RNN is presented. It receives an input as well as, recurring to the loop, the previous outputs and generates the current output. On the right side, the same RNN is presented but with its loop unrolled over three time steps. Each of the neural networks actually correspond to the same one but in a different time step, i.e. the first depicted RNN passes O_0 to itself when receiving I_1 as an input on the following

time step and so on. It is important to note that each line carries an entire vector from the output of a particular node to the input of another.

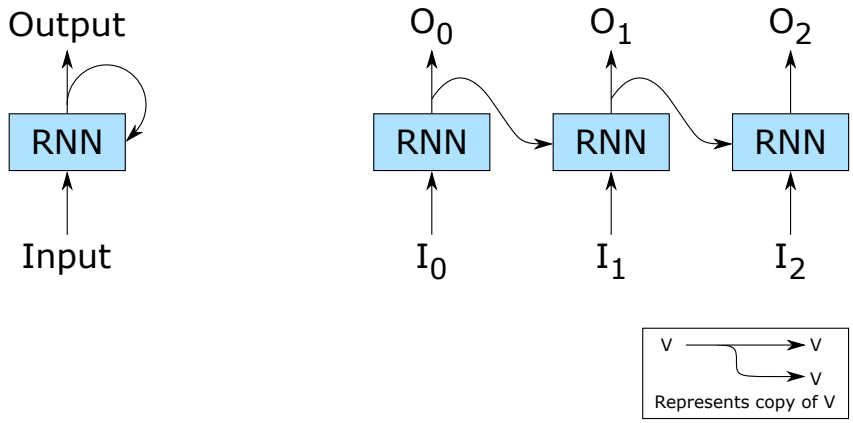

Figure 2.1: Recurrent Neural Network

RNNs have a clear difficulty when the time steps increase: they fail to maintain the representation of inputs from time steps much far behind, which means RNNs have trouble dealing with longer sequences. For instance, a RNN language model may be able to predict that the word "blue" should be the next word in the sequence "The sky is". However, when the gap between the relevant information and the place that it is needed is large, the same RNN language model might have trouble predicting words. Given the sequence "My parents are Portuguese but I was born in France, so I have multiple", it might have trouble predicting that "nationalities" is the next word. As a way to tackle this problem, Long Short Term Memory networks were introduced.

2.1.2 Long Short-Term Memory Networks

The LSTMs were introduced by Hochreiter in 1997 [14] and are a kind of RNNs that are capable of learning long-term dependencies. Similarly to the RNNs, the LSTMs have the form of a chain of repeating neural network modules when unrolled. A LSTM cell, depicted in Figure 2.2, has three different gates: the forget, the input and the output gate.

At a particular time step t, the forget gate, that corresponds to the sigmoid layer (a), decides what information is going to be thrown away from the previous cell state, c_{t-1}, which corresponds to the horizontal line represented at the top of the Figure 2.2. As such, (a) looks at the previous output value h_{t-1} and at the input x_t and outputs f_t, a vector containing a number between 0 and 1 for each dimension in the cell state c_{t-1}. A value of 1 is used to indicate that the corresponding information is to be kept while 0 is used to indicate that it is to be deleted.

Then it is necessary to decide what information is going to be added to the cell state and this is done in two parts. Firstly, the input gate, that corresponds to the sigmoid layer (b), decides which values are going to be updated and generates i_t. Secondly, the

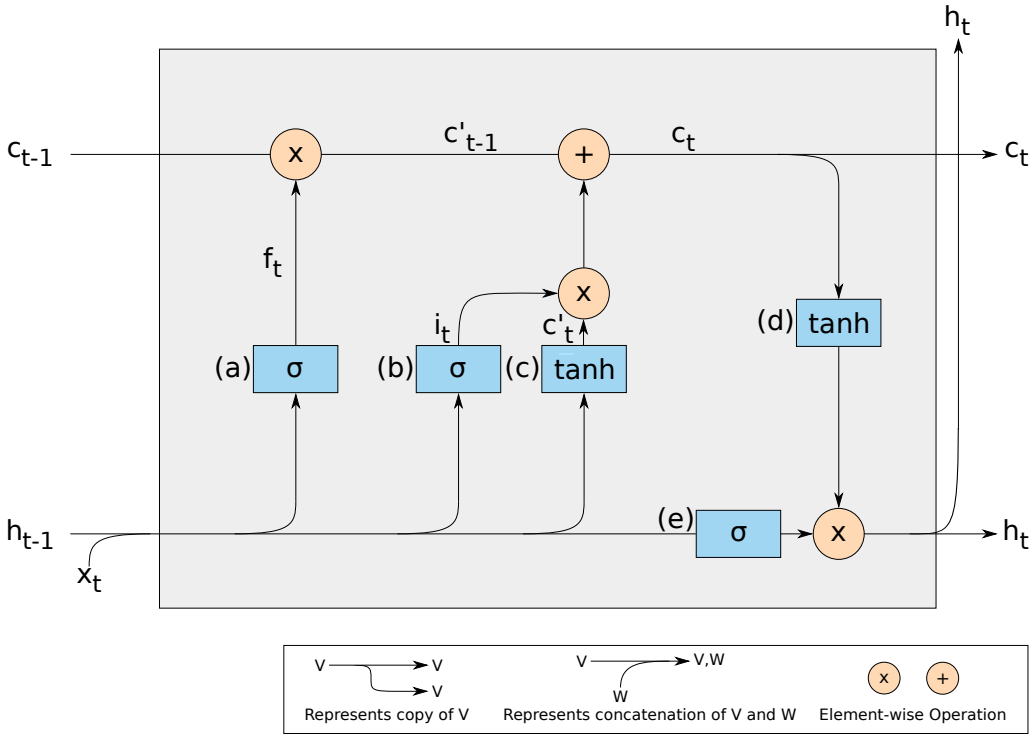

Figure 2.2: Long Short-Term Memory Network Cell

hyperbolic tangent layer (c) creates a vector, $c_t^{'}$, of new candidate values to be added to the cell state.

The vector $c_{t-1}^{'}$ is obtained by doing a element-wise multiplication between the previous cell state c_{t-1} and the forget vector f_t. Another similar multiplication is done between $c_t^{'}$ and i_t and the resulting vector is added with $c_{t-1}^{'}$ to obtain the new cell state, c_t.

Finally, the output gate, that corresponds to the sigmoid layer (e), decides what parts of the cell state, c_t, is going to be the output, h_t. The cell state c_t is run through a hyperbolic tangent function (d) and multiplied by the output of (e), that takes h_{t-1} and x_t as inputs. The result of this multiplication corresponds to the output value h_t.

LSTMs have achieved better results than plain RNNs but they have three times more parameters to be learned in a single cell. When using bigger networks, the training time and memory requirements significantly increases compared to the time and memory it takes to train plain RNNs. A faster alternative to LSTMs was introduced with the GRU.

2.1.3 Gated Recurrent Units Networks

GRUs were introduced by Chung et al. in 2014 [15] and a GRU cell is depicted in Figure 2.3. Unlike LSTMs, they do not have a cell state and instead of 3, have 2 gates: the reset and update gates.

The GRU's reset gate, that corresponds to the sigmoid layer (a) decides how much past information should be forgotten. As such, (a) takes as input the previous output

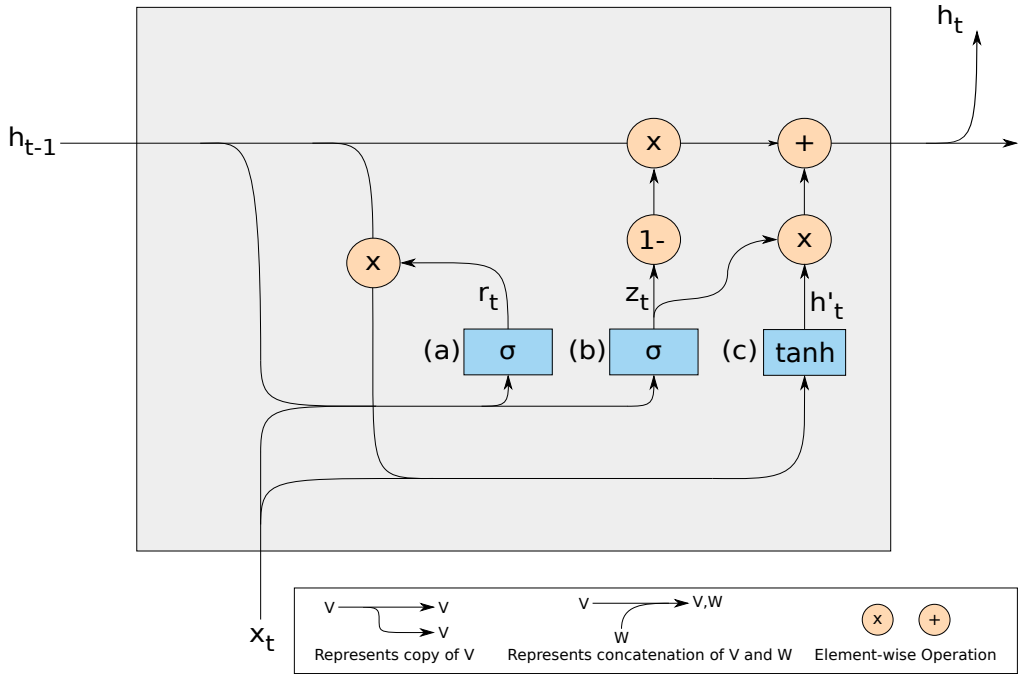

Figure 2.3: Gated Recurrent Units Network Cell

value h_{t-1} concatenated with the input x_t and outputs a vector r_t with values between 0 and 1. Then, this vector enter an element-wise multiplication with h_{t-1}.

The GRU's update gate, that corresponds to the sigmoid layer (b), acts similarly to the LSTM's forget and input gates. It decides what information should be deleted and what new information should be added. As such and similarly to a step of the reset gate, (b) takes as input the previous output value h_{t-1} concatenated with the input x_t and outputs vector z_t with values between 0 and 1.

By concatenating the result of the element-wise multiplication of the reset gate and x_t, a hyperbolic tangent function is fed and outputs h'_t.

Finally, h'_t and z_t are multiplied as well as h_{t-1} with $1 - z_t$. This latter multiplication transforms the values near 0 of z_t into values near 1 and vice versa, which leads to a forgetfulness of the dimensions with high values of h_{t-1}. Both results are added to generate the output value h_t.

On the one hand, GRUs are faster than the LSTMs since they have fewer parameters to learn. On the other hand, they perform worse than LSTMs. GRUs are used when it is necessary to train faster or do not have enough computation power at hand as they are a compromise between the speed of plain RNNs and the performance of LSTMs.

2.1.4 Bidirectional Recurrent Neural Networks

BRNNs were introduced by Schuster in 1997 [16] and aim to solve the problem of a word having more than one meaning, depending on the words that follow it. For example, the

word “bank” has different meanings in the sentences “Go to the bank to make a deposit” and “To reach the bank he had to swim a lot”. Figure 2.4 depicts a BRNN.

BRNNs are trained with similar algorithms as RNNs and solve this problem by having two hidden layers with connections running in opposite directions, allowing them to receive information both from the past and future states. When training, the left-right A and right-left A' states are processed by going through an input sequence from left to right and from right to left, respectively. Then, the results of these RNNs are concatenated and given as the output of the neural network.

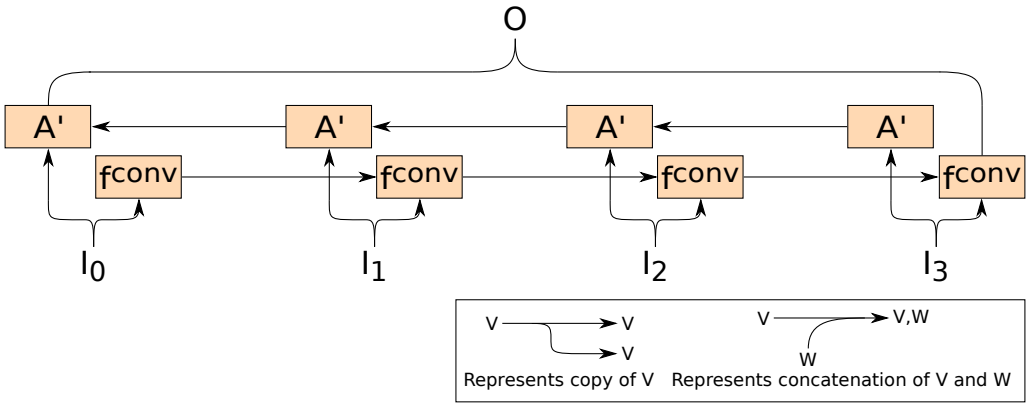

Figure 2.4: Bidirectional Recurrent Neural Network

2.2 Sequence to Sequence Models

This section presents an introduction to Sequence to Sequence models, their overall architecture and a commonly used optimization mechanism. Then, more advanced models that did not directly influence the dialogue systems reproduced in this dissertation are discussed as they are interesting in their own right.

2.2.1 Encoder-Decoder

An Encoder-Decoder model, Sequence to Sequence model or Seq2Seq model, is a subset of sequence models that takes as input a sequence of tokens and generates also a sequence of tokens as output. A representation of an abstract Encoder-Decoder model is depicted in Figure 2.5.

This model has an encoder and a decoder as its two main components and both are RNNs, commonly LSTMs. The task of the encoder network is to create a smaller and fixed size dimensional representation, h, of the input sequence. This representation is then forwarded to the decoder network which generates an output sequence.

Seq2Seq models are commonly used in automatic translators, where the input corresponds to a sentence and the output corresponds to its translation in another language.

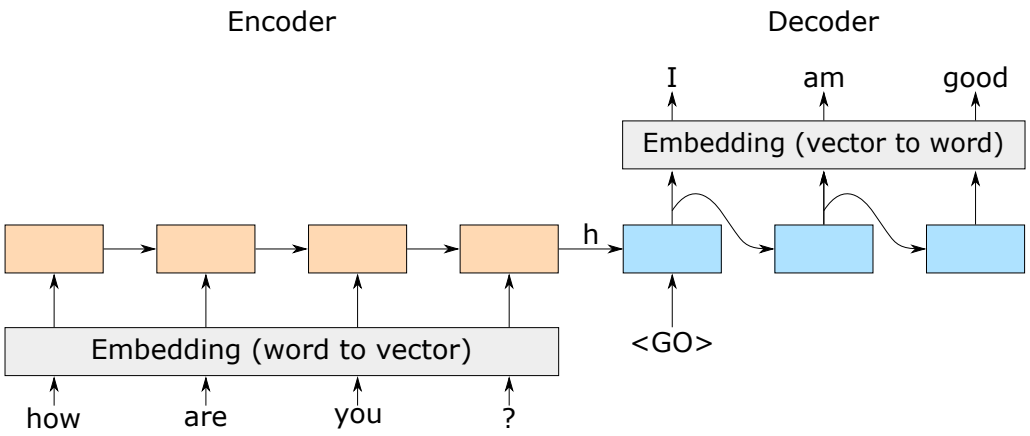

Figure 2.5: Encoder-Decoder

These models are also used for dialogue generation but instead of the output being the translation of a sentence, it is the reply to a given utterance.

Apart from these two components, many attempts to improve the performance of these models have lead to other components. One of the most commonly used components is the attention mechanism.

2.2.2 Attention Mechanism

Models that use RNN encoders and decoders have trouble keeping track of long-term dependencies and handling long sequences because the basic encoder-decoder architecture encodes the input sentence into a single fixed-length vector. When fed with a long sentence, this architecture often weakens the representation of the first part once it completes processing the whole input.

The attention mechanism was proposed by Bahdanau et. al. in 2014 [17] to alleviate this problem by helping the model to deal better with long source sentences. With attention, the model uses only parts of the input sequence where the most relevant information is concentrated to predict the next output word.

Attention is applied to the decoder component of the architecture and, instead of using a single context vector representing the input sentence, a separate context vector is used for each target word. These context vectors are calculated as a weighted sum of the encoder vectors of the words in the input sequence by an alignment model, which can be a feed-forward neural network, for instance. The weight assigned by the alignment model to each word vector reflects the importance of the relevant words in deciding the next state and in generating the output word at stake.

Although initially designed for neural machine translation, the attention mechanism is now used in various other tasks like image captioning and chatbots.

2.2.3 Learning Dialogue with Multiple Answers

In 2018, Rajendran et al. [18] published a paper that tackled the problem most systems have during training, that only one answer is considered to be right and therefore other options that may be equally plausible are dismissed.

A new method, Mask-memN2N, that can handle multiple answers was proposed. It has two phases: one in which the dialogue system tries to learn how to perform dialogue from the dataset by trying to mimic it, using supervised learning; and the other in which it learns to perform dialogue through trial and error, using reinforcement learning.

To mitigate the problem mentioned above, during the supervised learning phase, when the system is trained to produce one utterance, it is allowed to use only parts of the dialogue state vector, which has all the information from the dialogue so far and is used for next utterance generation or retrieval. This ability allows only parts of the network that were responsible for the prediction of that particular answer to be affected and the dialogue system can retain other distinct parts and values of the state vector. This is achieved by generating a mask vector m which decides which parts of the state vector s should be used for producing that particular utterance.

A new dataset was also introduced in this paper by applying some modifications to the dialogue from the bAbI project of Facebook AI Research [19], a project that is organized towards the goal of automatic text understanding and reasoning, and therefore creating permuted-bAbI dialogue tasks, which was proposed to be a testbed for goal-oriented dialogue tasks.

The paper concluded by comparing Mask-memN2N to Weston et al.'s [20] Memory Networks and Sukhbaatar et al.'s [21] End-to-end Memory Networks, which have been successful on various NLP tasks and perform well on original bAbI dialogue tasks. Mask-memN2N is able to handle multiple correct next utterances present in permuted-bAbI dialogue task better than these baseline models.

2.3 Variational Autoencoder Models

Autoencoders were introduced in 1987 by Ballard [22] in the context of machine vision and consists of an encoder, a decoder and the reconstruction loss function. Figure 2.6 depicts this model.

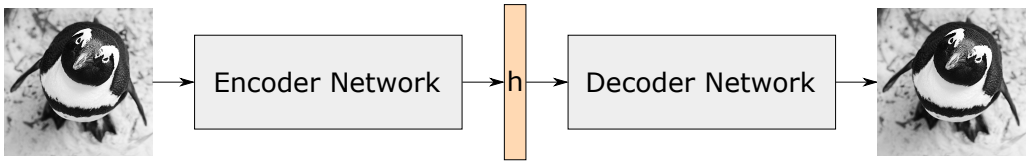

Figure 2.6: Autoencoder

An Autoencoder is a model that learns how to compress and encode data into a fixed-size vector h, so it can be later reconstructed by the decoder, as close to the original as possible, from this reduced encoded representation. The reconstruction loss function measures how well the Autoencoder is performing and how close the output is to the original input.

Variational Autoencoders (VAEs) were introduced in 2013 by Kingma et al. [23] and Rezende et al. [24]. They extend Autoencoders by introducing a variational mechanism that enforces similar hidden representations for similar inputs and supports powerful generative models that can obtain state-of-the-art results in image generation and reinforcement learning. Due to their capability of generating new data similar to the ones they were trained on, VAEs have also recently been used in dialogue generation systems as a way of generating more diverse results, i.e. as a way of generating new and unique sentences that are not present in the training dataset.

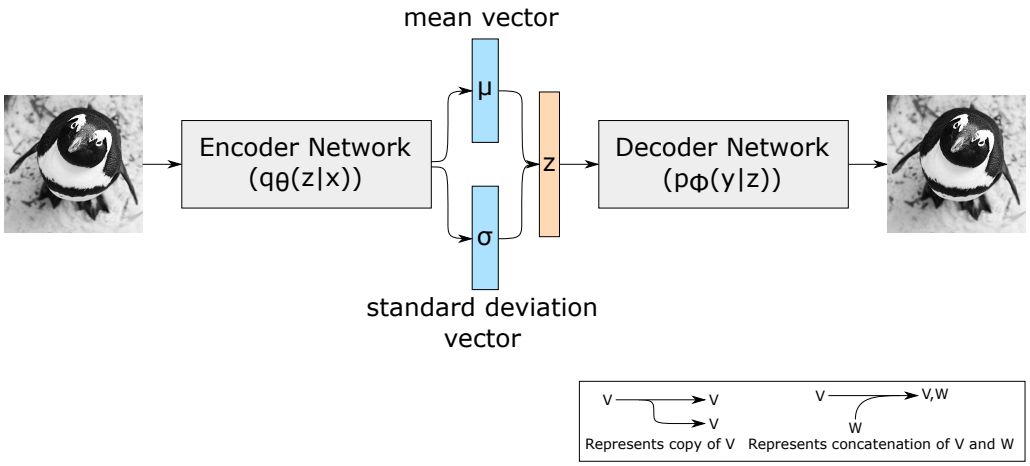

Figure 2.7: Variational Autoencoder

Similarly to a plain Autoencoder, the VAE consists of an encoder, a decoder and a loss function as illustrated in Figure 2.7. For the encoder, the notation $q_\theta(z|x)$ is used, where θ is the encoder's weights and biases, x is a datapoint that it receives as input and z is a compressed representation of x, the output hidden latent variable. This variable follows a Gaussian distribution with parameters μ and σ, which are approximated by the encoder network. For the decoder, the notation $p_\phi(y|z)$ is used.

Let us assume x is a 576-dimensional object, a 24 by 24-pixel photo. z is a compressed representation of this photo, i.e. it is smaller than a 576-dimensional object, generated by the encoder and it can be used to feed the decoder that is capable of reconstructing x when receiving this variable as input. In a simple Autoencoder, we could only obtain z by directly encoding it from a image but as we are trying to build a generative model and our model needs to be able to generate data on its own, we need a new way that does not rely on a new input, in this case x, to generate this variable. In order for this to happen, we add a constraint on the encoding network that forces it to generate latent vectors that roughly

follow a Gaussian distribution. More specifically, we feed x to a multilayer perceptron (MLP) that has the hyperbolic tangent function as its activation function and that, with its weights and biases, outputs the mean and standard deviation of a Gaussian distribution. By sampling this distribution, we can obtain new latent vectors, z, and consequently new outputs from a single input object.

In this model, since we are going from a smaller to a larger dimension (from the hidden representation, z, to the data x), some loss of information is expected to happen and we can measure it using the reconstruction log-likelihood $logp_\phi(x|z)$, which tells us how effectively the model learned to reconstruct x given its latent representation z. The loss function is the negative log-likelihood with a regularizer and is shown in Equation 2.1, initially presented in the original Variation Autoencoder paper [24].

$$l(\theta, \phi) = -E_{z\sim q_\theta(z|x)}[logp_\phi(x|z)] + KL(q_\theta(z|x)||\mathcal{N}(0,1)) \tag{2.1}$$

The first term of the right hand side of the equation is the reconstruction loss, or expected negative log-likelihood of the x datapoint, which encourages the decoder to learn to reconstruct the data.

The second term is a regularizer, the Kullback-Leibler divergence between $q_\theta(z|x)$ and $\mathcal{N}(0,1)$. This divergence measures how much information is lost when using q_θ to represent p. If the encoder outputs μ and σ vectors that are too far from 0s and 1s respectively, i.e. those from a standard normal distribution, it will receive a penalty in the loss. This regularizer keeps the encoder from cheating by giving each datapoint a representation in a different region of Euclidean space, discouraging it to attribute very different representations to similar objects by penalizing this behavior. This leads to more meaningful latent variables.

Chapter 3

Reproduced State-of-the-art Models

This section presents the models that were reproduced and are compared in Chapter 5. These models are explained in chronological order and the understanding of one is crucial to the understanding of the next since they share concepts among each other. Because these are probabilistic models, it is impossible not to include mathematical notation but most of these are, hopefully, well-explained throughout the sections in which they appear.

All of these models were implemented by their respective authors in Python, a popular programming language. Tensorflow and Pytorch are two Python open-source frameworks developed by two tech giants, Google and Facebook, respectively, and were used for these models (HRED was implemented using Tensorflow while VHRED and VHCR were implemented using Pytorch).

3.1 Hierarchical Recurrent Encoder-Decoder

In 2015, Sordoni et al. proposed the Hierarchical Recurrent Encoder-Decoder(HRED) [25] model, a model that extends the encoder-decoder architecture to the natural dialogue context. Figure 3.1 details HRED's recurrent architecture, unrolled over a dialogue composed of three turns. The HRED model consists of three GRUs: an utterance encoder (depicted on the bottom of the figure), a context encoder (in the middle) and a utterance decoder (at the top). Assume that $u_1, ..., u_n$ is a conversation consisting of n utterances, where $u_n = (u_{n,1}, ..., u_{n,m})$ is the n'th utterance and $u_{n,m}$ is its m'th discrete token.

HRED predicts the next utterance in a dialogue given the ones previously submitted by the user. The submitted utterances history is considered as a sequence at two levels: one at the word level and the other at the utterance level. For instance, let us say we have got a two turn conversation where the utterances are "At what time does Johnny arrive?" and "Johnny arrives at 9". At the word level, this conversation could be represented as [["At", "what", "time", "does", "Johnny", "arrive", "?"]["Johnny", "arrives", "at", "9"]]; at the utterance level, it could be represented as ["At what time does Johnny arrive?", "Johnny arrives at 9"].

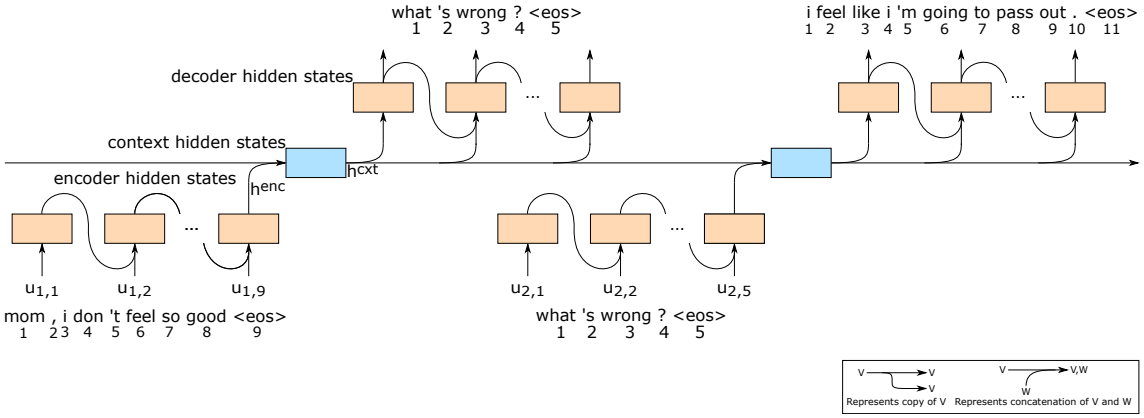

Figure 3.1: Hierarchical Recurrent Encoder-Decoder

Firstly, the utterance encoder GRU, f_θ^{enc}, encodes the previous utterance u_{n-1} into a fixed-size encoder vector h_{n-1}^{enc}. Secondly, the higher-level context encoder GRU, f_θ^{cxt}, takes h_{n-1}^{enc} and updates its hidden state, which correspond to the context vector h_{n-1}^{cxt} and is a summary of the past $n-1$ input utterances. Thirdly, h_{n-1}^{cxt} is used by the utterance decoder GRU, f_θ^{dec}, to predict the next utterance u_n.

While training, the parameters of the utterance encoder and utterance decoder GRUs are kept unchanged for every utterance in a dialogue, which helps the model to generalize across utterances.

HRED is not much different from the encoder-decoder model but the addition of the context encoder GRU hidden state to condition the predictions makes it achieve better results.

3.2 Variational Hierarchical Recurrent Encoder-Decoder

The Variational Hierarchical Recurrent Encoder-Decoder [26], or VHRED, was proposed by Serban et al. in 2016 as a solution to alleviate the problem that the HRED model has which consists in only having one source of variation, modeled through the conditional output distribution, for the output utterances. This imposes a strong constraint on the generation process and affects the model's ability to generate more meaningful dialogue utterances. The VHRED model's architecture is depicted in Figure 3.2.

Similarly to the addition of a normally distributed latent variable in the Variational Autoencoder, VHRED introduces a normal distributed latent variable into the HRED structure before the decoder component.

Both the HRED and the VHRED models contain the same three components (utterance encoder GRU, context encoder GRU and decoder GRU) and both use the utterance and context encoder GRUs in a similar fashion.

In VHRED, the context vector h_{n-1}^{cxt} is used to feed a MLP, a two-layer feed-forward neural network that has the hyperbolic tangent as its activation function. With its weights

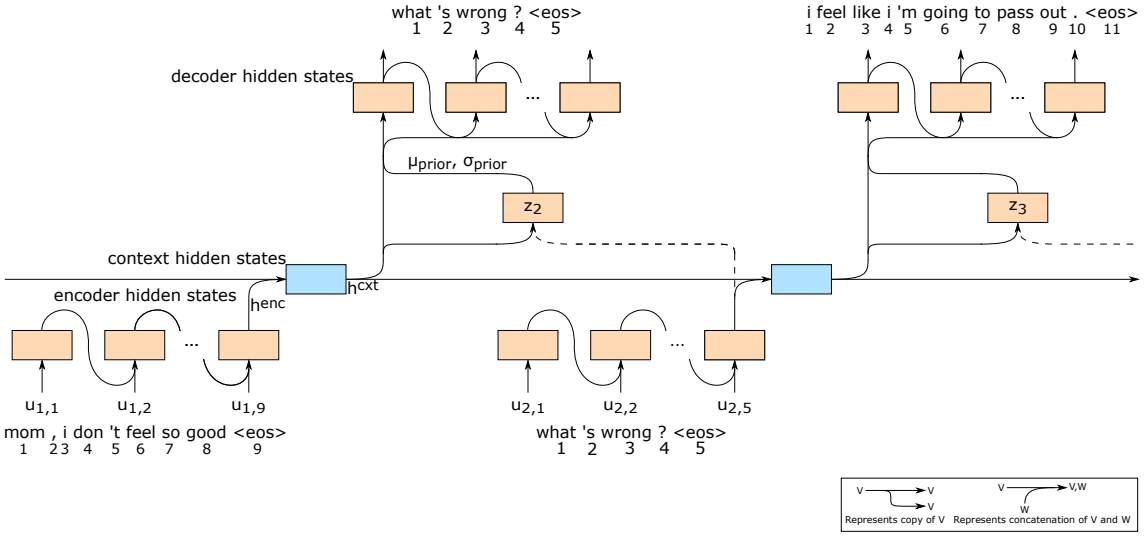

Figure 3.2: Variational Hierarchical Recurrent Encoder-Decoder

and biases, the MLP outputs the mean, μ_{prior}, and standard deviation, σ_{prior}, of a Gaussian distribution that corresponds to the approximate prior distribution P_θ. With this distribution, we can sample new latent variables to generate new data, like in the Variational Autoencoder model explained in Section 2.3.

However, unlike the VAE, instead of a loss function based on the reconstruction loss and KL divergence between $\mathcal{N}(0,1)$ and $q_\theta(z|x)$, here the model is optimized by maximizing the variational lower-bound expressed in Equation 3.1. In the first turn of the equation, $KL[Q||P]$ is the Kullback-Leibler (KL) divergence between distributions Q and P. The KL divergence calculates how different the probability distributions it receives as inputs are. Q_ψ (Equation 3.2) is the approximate posterior distribution, which is calculated in a similar way as P_θ (Equation 3.3) but also taking the current utterance u_n into consideration. In the second, we calculate the amount of information we received given the n utterance. The objective of this variational lower-bound is to balance the modeling powers of the decoder and the latent variable, instead of allowing any of these to become too powerful (which ultimately will lead to worse results).

$$\begin{aligned} logP_\theta(u_1,...,u_n) \geq \sum_{n=1}^{N} & -KL[Q_\psi(z_n|u_1,...,u_n)||P_\theta(z_n|u_1,...,u_{n-1})] \\ & + \mathbb{E}_{Q_\psi(z_n|u_1,...,u_n)}[logP_\theta(u_n|z_n,u_1,...,u_{n-1})], \end{aligned} \tag{3.1}$$

$$Q_\psi(z_n|u_1,...,u_n) = \mathcal{N}(\mu_{posterior}(u_1,...,u_n), \Sigma_{posterior}(u_1,...,u_n)) \tag{3.2}$$

$$P_\theta(z_n|u_1,...,u_{n-1}) = \mathcal{N}(\mu_{prior}(u_1,...,u_{n-1}), \Sigma_{prior}(u_1,...,u_{n-1})) \tag{3.3}$$

At training time, the sample z_n, drawn from the posterior distribution Q_ψ, is used to

estimate the gradient of the variational lower-bound given by Equation 3.1. The approximate posterior is parametrized by its own one-layer feed-forward neural network, which takes as input the output of the context RNN at the current timestep, as well as the output of the encoder RNN for the next utterance.

At test time, the sample z_n, drawn from the prior distribution P_θ for each sub-sequence, is concatenated with the output of the context RNN and given as input to the decoder RNN which then generates the sub-sequence token-by-token.

3.2.1 Degeneration Problem

A vanilla implementation of VHRED suffers from the degeneration problem which consists of setting the posterior $q(z|x)$ equal to the prior $p(z)$, making the KL divergence term assume a neutral value and learning to ignore the latent variable z. This will eventually make this model behave as a simple RNN model as it can achieve likelihoods that are close to optima simply by expressing arbitrary distributions over the output sentences. The need to force the model to use the global latent variable to achieve good likelihoods is therefore presented. Bowman et al. proposed in 2016 [27] two mechanisms to mitigate this issue: the KL cost annealing and the Word drop mechanism.

In the KL cost annealing approach, a variable weight is added to the KL term in the cost function at training time. This variable is initiated with the value of 0, to allow the model to learn to encode as much information in the latent variable as it can and then, as training progresses, its value is gradually increased until it reaches 1. This variable forces the model to smooth out its encodings and pack them into the region of the embedding space that is assigned a reasonably high probability by the Gaussian prior.

The word drop mechanism is a mechanism that weakens the decoder by removing some or all of the conditioning information during the training phase. This is achieved by randomly replacing some fraction of the conditioned-on word tokens with the generic unknown word token (*UNK*, for example).

3.3 Variational Hierarchical Conversation RNNs

The VHCR model was introduced by Park et al. in 2018 [28] and is a variational autoencoder model that was developed with the main goal of solving the degeneration problem that VAE conversation models have: decoders can learn to ignore latent variables and eventually are reduced to vanilla RNNs. This problem was first introduced in Section 3.2.1. The degeneration problem also happens because the hierarchical RNN can easily overfit to the training data by memorizing the context-to-utterance relation without relying on latent variables since frequently there only exists very few target utterances when conditioned on the context.

To address this problem, VHCR's key ideas are using a hierarchical structure of latent variables and exploiting an utterance drop regularization. This utterance drop regularization is similar to the word drop one described in Section 3.2.1 but instead of only weakening the lower-level decoder RNNs, the utterance drop depresses the hierarchical RNN decoders as a whole with a defined amount of probability. The architecture of VHCR can be seen as an extended version of VHRED and is depicted in Figure 3.3.

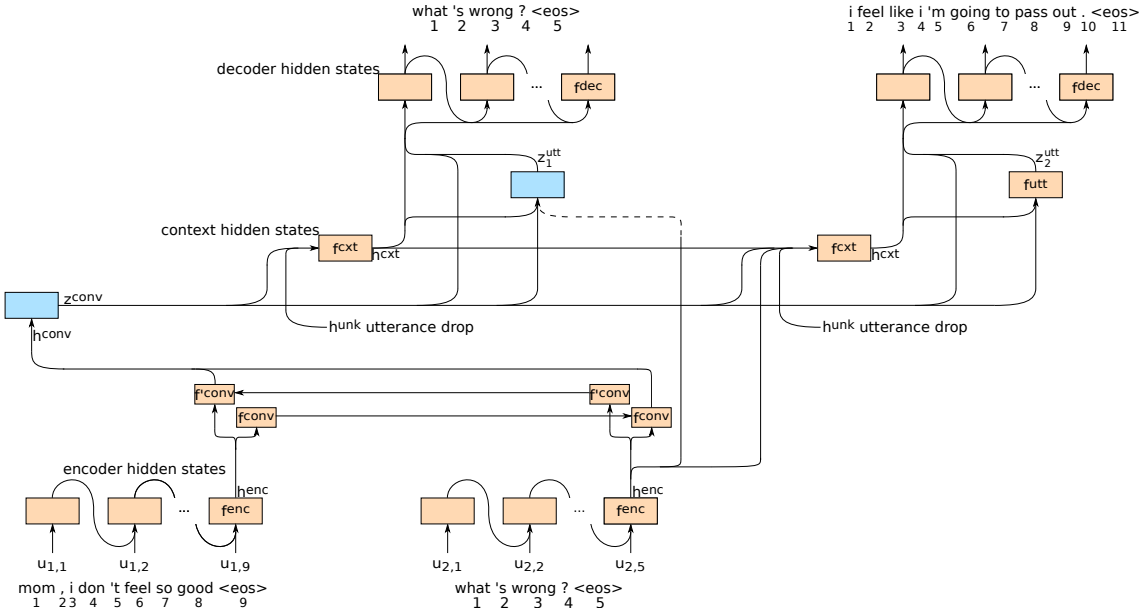

Figure 3.3: Variational Hierarchical Conversation RNNs

Similarly to the previous explained models, the first step of this architecture is to encode an utterance u_t using the encoder RNN, f_θ^{enc}, which will generate the encoder vector h_t^{enc}.

A global conversation latent variable, z^{conv}, was introduced in VHCR. This variable is inferred by using a bidirectional RNN denoted by f^{conv}. This RNN encodes all h_t^{enc} vectors of all utterances of the conversation so far, and generates a conversation encoder vector, h^{conv}. h^{conv} is then fed to a MLP and with its weights and biases, outputs a mean and a standard deviation of a Gaussian distribution that is going to be used to sample z^{conv}.

Subsequently, if $t = 0$, i.e. when the conversation thus far has only one turn, the context RNN f_θ^{cxt} is fed with z^{conv} and it outputs the context vector h_t^{cxt}. In the following iterations, i.e. when the value of t is greater than 0, the context vector is also obtained by the context RNN, which receives as inputs the previous context vector, h_{t-1}^{cxt}, the previous encoder vector, h_{t-1}^{enc} and the global conversation latent variable z^{conv}.

The next step is to obtain the approximate prior distribution P_θ. This is achieved by feeding yet again other multi-layer perceptron with the current context vector h_t^{cxt} and the global conversation latent variable, z^{conv}, which will generate the current mean μ_t and

standard deviation σ_t of the Gaussian distribution θ.

For the approximate posterior distribution Q_θ, we follow a similar approach as for the prior only this time, aside from the current context vector h_t^{cxt} and the global conversation latent variable, z^{conv}, the next encoder vector is also fed to the multi-layer perceptron.

Similarly to what was done in VHRED, one can quantify the degree to which this model learns global features by looking at the variational lower bound objective. The model's utterance latent variables, z_t^{utt}, are inferred by maximizing this variational lower-bound (Equation 3.4), where $KL[Q||P]$ is the Kullback-Leibler (KL) divergence between distributions Q_ϕ and P_θ.

$$\begin{aligned} logP_\theta(u_1,...,u_n) \geq \sum_{n=1}^{N} -KL[Q_\phi(z_n^{utt}|u_1,....,u_n,z^{conv})||P_\theta(z_n^{utt}|u_1,...,u_{n-1},z^{conv})] \\ + \mathbb{E}_{Q_\phi(z_n^{utt}|u_1,....,u_n,z^{conv})}[logP_\theta(z_n^{utt}|u_1,...,u_{n-1},z^{conv})], \end{aligned} \tag{3.4}$$

Finally, z_t^{utt} is concatenated with z^{conv} and h_t^{cxt} and fed to the decoder RNN f_θ^{dec} which then generates the sub-sequence x_t word-by-word.

As previously mentioned, at each time step and as a way to weaken the autoregressive power of the hierarchical RNN, the utterance encoder vector h_t^{enc} is randomly replaced with a generic unknown vector h^{unk} with a probability p. By inducing noise into the context vector h_t^{cxt}, the data sparsity problem is alleviated.

3.4 Online Chatbot Demonstration

As my main goal was to train a model so it could be used as a chatbot, a program to handle this using the three model's architectures was developed. As such, the program *chat.py* is able to interact with the reproduced models and in a time effective way, serves our purpose. A freely online service based on this program is available[1] and its front page is depicted in Figure 3.5.

An interaction with this program starts with the hard-coded greeting utterance "Hello there, how can I help you?". This utterance has the objective of informing the user that the program is expecting to receive an utterance as input, which is going to be answered by it.

After the user has given his utterance as input, the program updates the conversation context, which is stored in three different variables. These variables contain information of all of the current dialogue's turns.

More specifically, the *Sentences* variable keeps track of the utterances of the conversation by having an array of arrays in which every entry corresponds to a particular word, the tag "<eos>" that determines the end of each utterance and, if the utterance length

[1] https://portulanclarin.net/workbench/lx/chatbot

is smaller than the one established as a network's hyperparameter, some padding tokens, "<pad>". This padding step takes place because the inputs are going to be processed in batches. If the utterance length is greater than the hyperparameter one, the utterance is truncated and the tokens at the end of the utterance are ignored.

"Sentence_length" keeps an array of integers that record the number of words expressed in each utterance, including the tag.

Finally, "conversation_length" keeps track of the number of turns that have happened in the corresponding conversation. An example of the state of these files of a 3 turn dialogue with a maximum utterance length of 15, which is too small for a real case but enough to demonstrate the example, is presented in Figure 3.4.

These three variables are then concatenated with two vocabulary files that the model generated during the training phase which contain the words it is able to recognize as well as with the batch size hyperparameter. The result is called a "data loader" and is used to feed a trained model, which outputs the response. When the model receives the utterance "Ok, thanks" as input, it returns the hard-coded utterance "you are welcome".

One useful aspect of this program is that it allows users to interact with different models. It is also possible for the user to simulate a whole conversation and receive the response of the chatbot to that given context. In order to do this, the user should input multiple utterances, each on a separated line that corresponds to a particular turn of the conversation.

It is worth mentioning that this program only receives an input, processes it and feeds a trained model to receive the output it generates. This means that no changes were made to any of the chatbot's architectures and as such, this program uses them as they were proposed on the respective author's papers.

Conversation_length

[3]

sentence_length

[[8, 3, 8]]

sentences

[[['when', 'will', 'ubuntu', '19.10', 'be', 'released', '?', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'], ['already', 'was', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'], ['i', 'am', 'talking', 'about', '19.10', 'not', '19.04', '<eos>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>']]]

Figure 3.4: Chatbot Context Variables

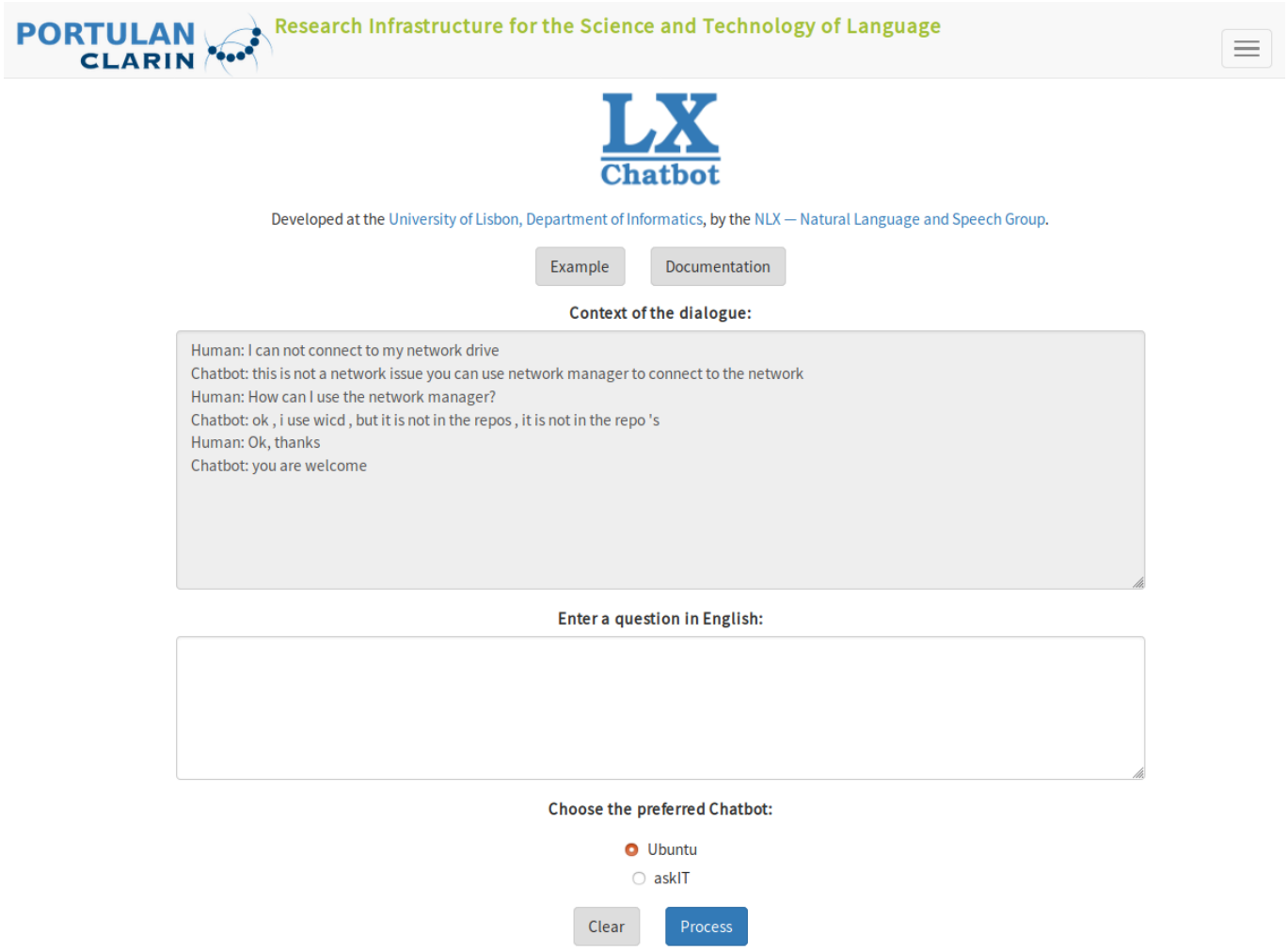

Figure 3.5: Online Chatbot Demonstration

Chapter 4

Datasets and Tools

Several advances have been made over the past decade in the areas of speech processing and language understanding and recent results suggest that data-driven approaches are promising and feasible, contrasting to hard-coded approaches.

Such data-driven approaches usually rely on massive datasets in order for the models to be trained, which has been leading to the creation of multiple large and public datasets that fit this need. In the next sections, some of the datasets that had a bigger impact in dialogue systems and subsequently, in this project, are presented. Not all of these datasets were used to train the models presented but they, at least, served as an inspiration for the work that was developed.

4.1 Ubuntu Dialogue Corpus

In 2015, Lowe et al. [29] introduced the "Ubuntu Dialogue Corpus", a dataset containing almost one million dialogues with multiple turns, with a total of over seven million sentences and one hundred million words. Table 4.1 summarizes some properties of this corpus and Figure 4.1 shows a sample dialogue from this dataset. These dialogues were extracted from the Freenode Internet Relay Chat (IRC) network's Ubuntu channel logs that were used to receive technical support for various Ubuntu-related problems. The dialogues from this corpus follow a few constraints such as: each dialogue corresponds to only one conversation between two humans; each dialogue contains at least three turns and has a task-specific domain.

In addition to this corpus, suitable algorithms for the model's evaluation were presented in that paper. These algorithms range from the TF-IDF metric, intended to reflect how relevant a word is to discriminate the topic of a document in a corpus, to more sophisticated neural models such as RNNs and LSTMs.

# dialogues (human-human)	930,000
# utterances (in total)	7,100,000
# words (in total)	100,000,000
Min. # turns per dialogue	3
Avg. # turns per dialogue	7.71
Avg. # words per utterance	10.34
Median conversation length (min)	6

Table 4.1: Properties of the Ubuntu Dialogue Corpus

2007-10-21T10:21:00.000Z	fabio		any one know a beter software than wine?
2007-10-21T10:22:00.000Z	fabio		gamming
2007-10-21T10:22:00.000Z	fabio		i have the wine
2007-10-21T10:22:00.000Z	fabio		and works good
2007-10-21T10:22:00.000Z	fabio		but in gta it slow
2007-10-21T10:23:00.000Z	fabio		cedega have one problem
2007-10-21T10:23:00.000Z	fabio		the textures
2007-10-21T10:24:00.000Z	fabio		well any one have the lastest version of cedega?
2007-10-21T10:24:00.000Z	Vlet	fabio	yeah, running wow in it now
2007-10-21T10:25:00.000Z	fabio		vlet
2007-10-21T10:25:00.000Z	fabio		can u pass to me┬┐
2007-10-21T10:25:00.000Z	Vlet		fabio
2007-10-21T10:25:00.000Z	Vlet	fabio	no
2007-10-21T10:25:00.000Z	fabio		ok
2007-10-21T10:25:00.000Z	fabio		hehehe
2007-10-21T10:25:00.000Z	fabio		im sory
2007-10-21T10:26:00.000Z	fabio		one more thing
2007-10-21T10:26:00.000Z	fabio		to instal the nvidia drivers
2007-10-21T10:26:00.000Z	Vlet	fabio	$5 is not hard to come by for gaming pleasure :)
2007-10-21T10:26:00.000Z	fabio		i can instal with evel?
2007-10-21T10:28:00.000Z	fabio	Vlet	see the pvt
2007-10-21T10:29:00.000Z	fabio		linux sucks
2007-10-21T10:29:00.000Z	fabio		hehe
2007-10-21T10:30:00.000Z	Vlet	fabio	okay, any other questions?

Figure 4.1: Example of an Ubuntu Dialogue Corpus dialogue

4.2 DSTC Corpora

The Dialog System Technology Challenge (DSTC) is an on-going series of research community challenge tasks. Since 2013, a number of challenges have been proposed yearly related to dialogue systems and for most of these challenges, a new dataset was released. However, not all of this dataset is freely available.

In the first three DSTC editions[30], datasets were provided and each is a medium-sized spoken dataset obtained from human-machine interactions with restaurant and travel information systems. While these datasets may not be particularly useful for chatbots due to their nature, they can be used in goal-oriented dialogue systems in which it is important to estimate the intentions of a user throughout a conversation.

While still being related to traveling, the DSTC4[31] dataset released in 2016 is a collection of 35 conversations between tourists and tour guides over Skype. Figure 4.2 shows a sample dialogue from this dataset. All the dialogues, with a total length of 21

hours, were manually transcribed and annotated with speech act and semantic labels for each turn. Both speech act and semantic labels can be used to better understand the text as these give additional information about the given sentences (for instance, if a sentence is an acknowledgement or if it is a recommendation about something). This dataset consists of discussions relating to hotels, flights and car rentals in Singapore.

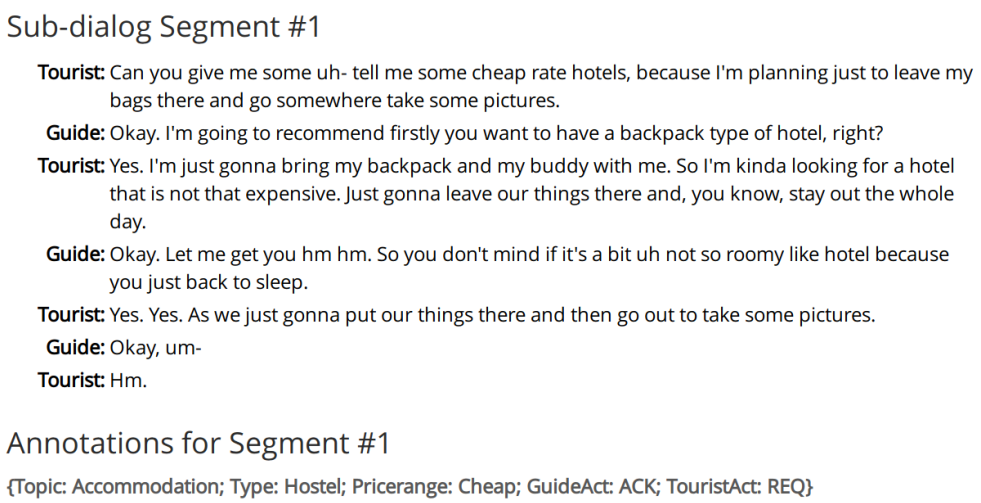

Figure 4.2: Example of a DSTC4 corpora dialogue and corresponding main task reference annotations

The DSTC5[32] dataset, released in 2017, comprises human-human spoken dialogues related to the same tourist information domain as DSTC4 and they were obtained in a similar fashion, however, this time not only one but two languages were included (English and Chinese).

4.3 Cornell Movie-Dialogue Corpus

Scripted corpora is also often used for dialogue systems. This can be obtained by extracting dialogues from movies or TV series. There are a few such corpora freely available on the internet but it is much easier to find unlabeled subtitle data than actual scripts where each utterance is properly tagged with the appropriate speaker.

The Cornell Movie-Dialogue Corpus[33] is one of the most well-known datasets of scripted corpora and Figure 4.3 shows a sample dialogue from this dataset.. It contains 220,000 dialogue excerpts but it only contains 300,000 utterances, which indicates that most of the excerpts consist of a single utterance. An interesting feature of this dataset is that it includes movie metadata such as IMDB[1] ratings and the character's and movie's

[1] an online database of information related to films, television programs, home videos, video games, and internet streams

```
L253836 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ Oh -- there's no real thing -- it doesn't exist.
L253835 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ I fell in love last night -- the real thing.
L253834 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ You're not a bit like you were yesterday.
L253833 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ I feel sorry for him.
L253832 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ Why?
L253831 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Then dance the next number with Kringelein.
L253830 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ Yes -- I'd love to.
L253829 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Would you like to make a man happy?
L253828 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ I'll do anything for you.
L253827 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Will you do me a big favor?
L253826 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ That was lovely.
L253816 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Yesterday -- yes -- that was yesterday.
L253815 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ You were very different yesterday.
L253814 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Money.
L253813 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ Chasing what?
L253812 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Chasing around.
L253805 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Sorry.
L253804 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ I'd given you up.
L253271 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ All right. We'll dance.
L253270 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ We'll dance.
L253269 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Really?
L253268 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ Of course.
L253267 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Yes? -- Tomorrow?
L253266 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ You're very funny --
L253265 +++$+++ u1184 +++$+++ m78 +++$+++ BARON +++$+++ Yellow Room where they dance --
L253264 +++$+++ u1188 +++$+++ m78 +++$+++ FLAEMMCHEN +++$+++ Where downstairs?
```

Figure 4.3: Example of Cornell Movie-Dialogue Corpus sentences

genre. This could be of particular interest for entertainment chatbots as it could be possible to create a model taking into consideration the various types of characters.

4.4 Developed Reddit Dataset Extraction Tool

One of my contributions is a tool that is able to collect real human to human interactions from the public website Reddit[2] as a way to build new datasets. This section details what Reddit is and what kind of content it contains, how can this content be used to create new datasets and how does this proposed tool work.

4.4.1 What is Reddit?

Reddit is a social news aggregation, web content rating, and discussion website, founded in June 2005. Registered members submit content to the site such as links, text posts and images, which are then voted up or down and discussed by other members. According to recent statistics, the platform has around 330 million monthly active users. As of now, according to Alexa [34], Reddit is, globally, the 13th most visited website, ranking 10th in Portugal.

Reddit is organized into around 1.2 million communities, known as "subreddits", each covering a different topic which makes it easy to find subreddits that are related to anything one might be interesting in. This might include casual chit-chat conversations (which can be found at *casualconversations*), sports discussions (which can be found at subreddits like *nbadiscussion*) or, of course, IT (which can be found at subreddits like *programming*).

[2] www.reddit.com

After a submission has been posted, users can post comments discussing the submission. These comments are organized in threads, which makes it easy to reply to a particular comment, even if it was a reply to another one, which also has more replies associated. Figure 4.4 exemplifies a typical Reddit submission with its comments and information about the subreddit in which it was submitted.

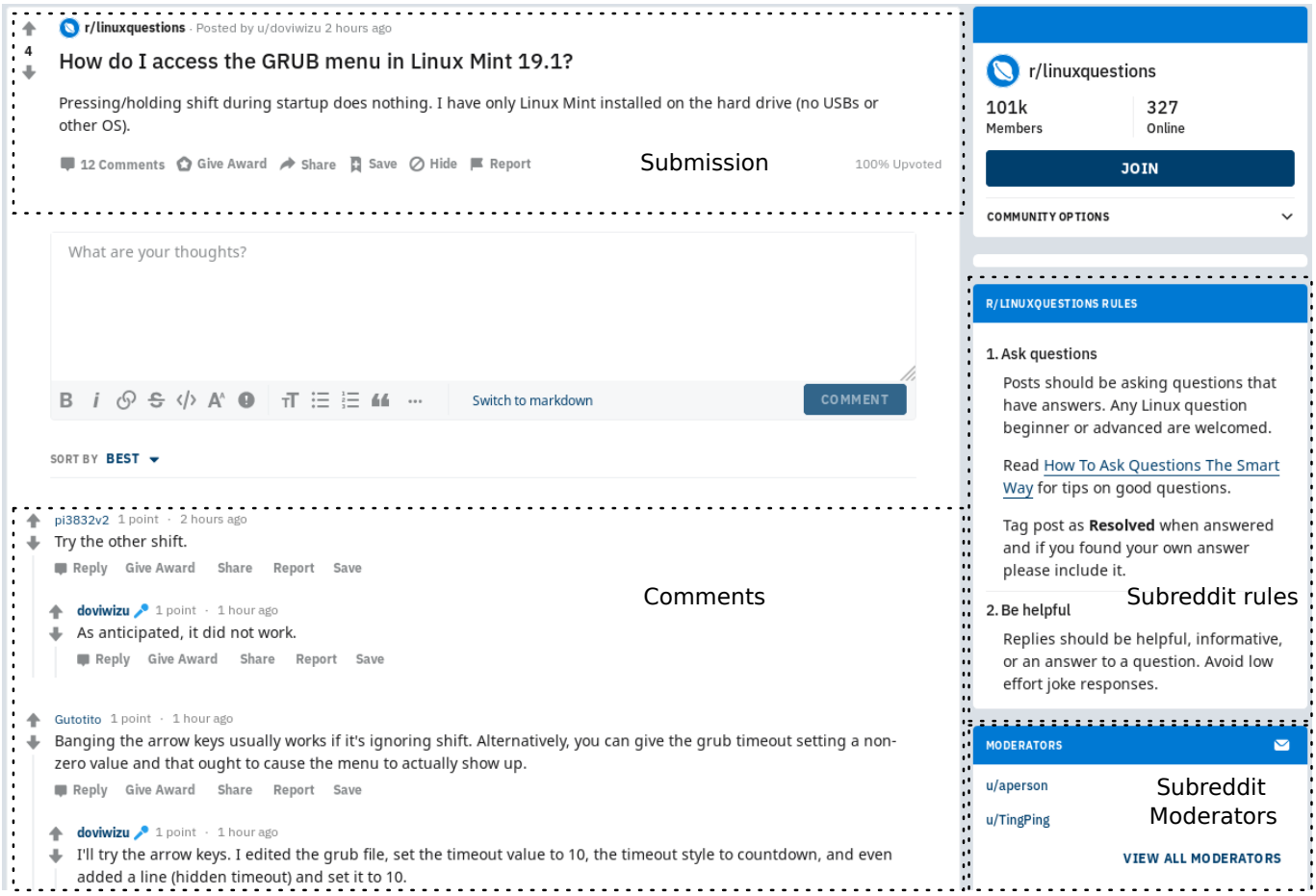

Figure 4.4: Reddit's Fields

Each subreddit is overseen by one or more moderators that make sure the content that is posted follows a predefined set of rules specific to that particular subreddit. For example, in the subreddit *guitar*, a general subreddit about guitars, every submission needs to have a tag in the beginning of the title for the submission to be considered. In this case, the following tags are allowed and they make it easy to see what kind of content a submission relates to: *[GEAR], [QUESTION], [NEWBIE], [PLAY], [OC], [DISCUSSION], [NEWS]*. These rules guarantee that unrelated or unwanted media is not posted and that the subreddit stays organized in a certain way, which is something that works in great favor for the tool presented in the next subsection. It is to be noted that the more popular the subreddit is, the higher the probability of having highly dedicated moderators.

Every public submission and corresponding comments are monthly scrapped into a

submission and comment databases, respectively, at Pushshift[35]. Pushshift is a website that contains various articles relating to big data, social media ingest and analysis and general technology trends.

With the tool presented in the next subsection and these databases, it is possible to extract dialogue datasets from any existing subreddit, i.e. it is possible to extract datasets for almost any subject. However, depending on which subreddits we choose to consider, the size of the resulting dataset may vary.

The main purpose of this tool is to aid researchers obtaining new, natural and mostly informal corpora, especially related to domains for which corpora are not easily found.

4.4.2 Reddit Dataset Extraction Tool

The Reddit Dataset Extraction Tool (RDET) I developed is divided in three different programs, *process_comments*, *process_submissions* and *extract_dialogues*. The first two mentioned programs extract database entries which can be either comments or submissions from the specified subreddits by the user, as further explained below. The *extract_dialogues* program takes these extracted entries and assembles them into dialogues. The Python module *redditcorpus* contains functions that are shared by all three programs.

This tool is able to extract dialogues from more than one subreddit at a time, which allows dialogues from multiple subreddits to be joined in the same dataset as a way to obtain one with a larger dimension. This is what was done for the creation of the askIT dataset as we will see in Section 4.5.

4.4.3 process_submissions and process_comments

The Reddit database dump is available from Pushshift's website[3] as previously noted and contains multiple JSON files corresponding to individual months. Comments and submissions, along other types of content such as information on moderators, are split up in different databases. Every database entry, whether it belongs to a comment or submission database, has a wide range of data and is not only composed by the body of text a particular user wrote on Reddit. However, comments and submissions do have different kinds of information that may be present. The list of fields that are kept by RDET is defined in the *redditcorpus* module and is shown in the Table 4.2. These fields may be used to filter the extracted dataset. For instance, we may want a dataset containing only comments that do not have more downvotes than upvotes.

It is important to note that the body of a submission or comment is not necessarily a sentence, as it may contain multiple sentences.

Another thing to note is that comment's parent IDs are in fact a composite field as their prefix indicates the type of content they refer to. Let us say we have a submission whose

[3] http://files.pushshift.io/reddit/

Comment	Submission
subreddit	subreddit
body	body
id	id
author	author
posting time	posting time
number of upvotes/downvotes	number of upvotes/downvotes
parent id	title
-	linked URL
-	adult content warning

Table 4.2: Reddit's Comments and Submission Fields

id is "1an3ou". A comment's parent id related to this submission could be "t3_1an3ou", i.e "t3" indicates that the parent is a submission. This comment may have as an id something like "2fn1fi" and a comment that is a reply to this one could have as id "t1_2fn1fi", meaning that the current comment is a reply to other comment, and not to a submission. A graphical representation of this example is shown in Figure 4.5.

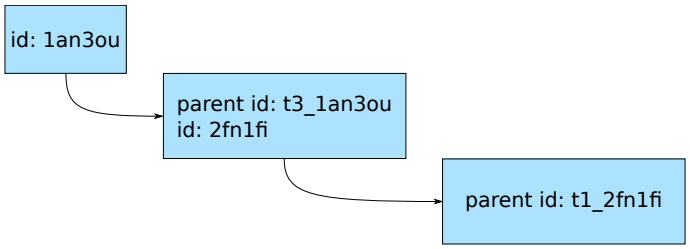

Figure 4.5: ID Trees

The purpose of *process_comments* and *process_submissions* is to extract the entries from the submissions and comments databases and keep only the data fields that we intend to keep, as not all data fields are useful for our purpose. These entries also need to have been posted on our subreddits of interest. In order to achieve this, these programs read the databases line by line and when such entries are found, the wanted fields are extracted and written to a file that has as file name, the ID of that submission or comment. This does not mean that a new file is always created every time an entry is written. If the corresponding file of an entry already exists, the entry is written to that existing file, which assures us that, when this step is over, any given file contains all entries related to it. A sample of a dialogue extracted with these programs is depicted in Figure 4.6.

4.4.4 extract_dialogues

The purpose of the *extract_dialogues* program is to use the files output by *process_comments* and *process_submissions* as input and output a file for each existing dialogue on these

```
{"id": "c93mghg", "parent_id": "t3_1b4pzi", "link_id": "t3_1b4pzi", "body": "Sim, prestem atenção
apenas aos bancos do Sul, enquanto os bancos bem lá do Norte, na Islândia, vos deixam sem cheta.
**LAWL**", "subreddit": "portugal", "score": -1, "ups": -1}
{"id": "1b4pzi", "title": "Oje - o Jornal Economico - Economia - Britânicos querem transferir
depósitos que estão nos países do sul", "url": "http://www.oje.pt/noticias/economia/britanicos
-querem-transferir-depositos-que-estao-nos-paises-do-sul", "subreddit": "portugal",
"is_self": false, "selftext": ""}
```


Figure 4.6: Sample of a dialogue extracted with the process_comments and process_submissions programs

files, i.e. it creates a dialogue from every possible complete thread of comments in every submission.

For each input file, this program recursively searches for child nodes by trying to match a particular entry id to other entry's parent id. When it reaches a node that does not have any child, the program generates a file that contains the dialogue up until the current point. Each dialogue has the concatenation of the title and body of a submission as well as each comment's body as turns. At this step, each turn is normalized by deleting existing line breaks.

These output files are placed inside folders that organize all the data and facilitate the task of finding dialogues about a particular topic. All generated files of a particular submission are stored in a folder that assumes the submission's ID as its name. In turn, these folders are placed inside folders that assume the subreddit in which the submission was posted as its name.

For clarification, let us assume we have a submission, S_1, that is composed of a title and a body. This submission, or post, has two reply comments, C_1 and C_2, that were written by two different users. C_1 has two replies, $C_{1,1}$ and $C_{1,2}$ while C_2 does not have any. In turn, $C_{1,1}$ also has two replies, $C_{1,1,1}$ and $C_{1,1,2}$. In this case, *extract_dialogues* is going to generate four different dialogues. These four dialogues will have the title of the submission concatenated with its body as the first dialogue turn (represented by S_1), which will be continued by the comments and resulting in the following threads:

- $S_1 - C_1 - C_{1,1} - C_{1,1,1}$;
- $S_1 - C_1 - C_{1,1} - C_{1,1,2}$;
- $S_1 - C_1 - C_{1,2}$;
- $S_1 - C_2$.

4.5 Extracted askIT Dataset

I constructed a dataset with RDET, the askIT dataset. This is a collection of dialogues extracted from the subreddits related to Information Technology (IT) listed in the second column of Table 4.3. The reason why these subreddits were chosen is because they are exclusively related to IT support, even though from various topics (column 1 of Table 4.3). Table 4.4 summarizes some properties of this corpus.

General Topic	Subreddit	# Followers	# Submissions	# Comments
General IT	TechnologyHelp	50	1	3
	computerhelp	1862	1164	5237
	AskTechnology	14317	11004	44868
	techquestions	104	7	28
	computer_help	3829	625	2680
	24hoursupport	16228	24079	114183
	TechnologyProTips	17571	6246	1553
Linux	linux4noobs	129550	63500	294595
	linuxquestions	101202	59993	273314
PC Gaming	pcgamingtechsupport	12325	11110	52775
Computer Science	AskComputerScience	34306	6322	26257

Table 4.3: askIT's Subreddit Properties

# dialogues (human-human)	179,358
# turns (in total)	820,186
# utterances (in total)	2,480,283
# tokens (in total)	61,842,638
Min. # turns per dialogue	3
Avg. # turns per dialogue	4.57
Avg. # utterances per turn	3.02
Avg. # tokens per utterance	24.93

Table 4.4: Properties of the askIT Corpus

Other subreddits could be taken into account (like the most popular IT related subreddit *programming*) but as these did not follow a Q&A structure, they did not align with what was intended for this dataset and therefore I opted not to include them.

Due to the support nature of the subreddits used for this dataset, none has the required tag rule mentioned in Section 4.4.1 and no preprocessing to take care of this was needed.

This dataset has 820186 comments in 179358 dialogues, meaning that each dialogue has on average 5 turns. However, it is worth mentioning that it is possible that some dialogues have a much larger number of turns and therefore, we can have a significantly larger amount of smaller dialogues.

Since there were no restrictions imposed on the year in which these dialogues happened, some information presented may be outdated or inaccurate now. Another thing to keep in mind is that there's no way of verifying that what users are saying is true, so by using this dataset to train a chatbot, it is assumed that the chatbot may be partially trained on wrong information. Hopefully, only a small part of the dataset will contain such errors.

# dialogues (human-human)	218,550
# turns (in total)	1,063,462
# utterances (in total)	2,303,672
# tokens (in total)	58,964,715
Min. # turns per dialogue	3
Avg. # turns per dialogue	4.87
Avg. # utterances per turn	2.17
Avg. # tokens per utterance	25.60

Table 4.5: Properties of the Portuguese Corpus

4.6 Extracted Portuguese Dataset

The Portuguese dataset is another dataset that was extracted with RDET, but the goal here was to extract a dataset in a language other than English. Table 4.5 summarizes some properties of this corpus. The *portugal* subreddit was chosen as a source since most content presented is in Portuguese. This subreddit contains dialogues related to a wide range of topics including discussions about politics and sports. However, there is some noise in the dataset as it is sometimes used by people who do not speak the language to ask questions about the country. No cleanup was performed to the dataset prior to the experiments presented in the next chapter and that cleanup should be addressed in future work as it might help models trained on this dataset achieve a better performance since they would not have to handle two distinct languages at once.

Since it is an informal and casual subreddit, curse words and offensive content may be present, which is not ideal for the training of chatbots. Either way, this dataset can be used to study how well does a chatbot handle dialogues not written in English, which are often the baseline, and covering a wide number of topics, instead of focusing on a singular one.

This dataset has 1063462 turns in 218550 dialogues, meaning that each dialogue has on average 5 turns. Again, some dialogues might have considerably more turns than others.

Chapter 5

Performance Evaluation

5.1 Dialogue Systems Evaluation

The evaluation of dialogue systems has been widely discussed by the scientific community and still does not have a fully satisfactory approach. This section describes a set of metrics that were first proposed in the context of Machine Translation and Summarization and have also been used to evaluate dialogue systems. However, as we will see below, these metrics are not perfectly suited for the evaluation of these systems and as such, these are of historical interest only in the context of this dissertation. The metrics that were used to evaluate the reproduced models, which are the model's perplexity and negative log-likelihood and embedding-based metrics, are presented in Section 5.4.

BLEU [36] is a metric that is able to measure how different are the model's output responses from the reference utterances. It compares contiguous sequences of tokens of the output with sequences of tokens in the reference utterances and in a weighted fashion counts the number of matches in a position independent way. The higher the score, the higher the similarity and therefore, the better the output quality. This score is commonly used to evaluate machine translation.

METEOR [37] is another metric used to assess the generated responses by aligning them to reference utterances. These alignments are based on exact, stem, synonym, and paraphrase matches between words and phrases.

ROUGE [38] is a set of metrics that are used for evaluating automatic summarization and machine translation models that count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans.

As mentioned, these metrics are not well suited for evaluating chatbots since they assume that a valid response to a given question has significant word overlap with the reference responses. In chatbots, where there is a large diversity in the space of acceptable responses, this overlap may very well not happen in plausible responses and therefore a good chatbot model may achieve bad results in this kind of evaluation. Figure 5.1 shows

Context of Conversation
Speaker A: Have you seen today's xkcd?
Speaker B: I can't, my browser keeps crashing!

Ground-Truth Response
Try rebooting your computer.
Model Response
Have you tried to reboot?

Figure 5.1: BLEU Score of 0 on a Valid Response

an example of a conversation in which the model outputs a valid response to the given context but still gets a BLEU score of 0.

5.2 Models and Datasets

All the HRED, VHRED and VHCR models were trained on the Ubuntu dataset, reproducing the experiments reported in the VHCR's paper, using the given training, validation and test sets. VHRED was trained resorting to the KL cost annealing and the Word drop mentioned in Section 3.2.1.

These models were also trained on the askIT dataset, as well as on the Portuguese dataset, to assess how comparable the results obtained with the extracted datasets are with respect to the results obtained with a well-known dataset. Both of these datasets were split into training, validation and test sets, containing respectively 80%, 10% and 10% of the corresponding dataset.

5.3 Pre-processing

Before training a model on a dataset, some pre-processing to that dataset is usually needed. For the extracted datasets mentioned in Section 4.5, the following operations were executed as a way to try to better standardize all the data.

Firstly, non-ascii characters were replaced by ascii alternatives. This includes characters we do not expect, and do not want, the model to learn and that if present on the dataset, it will only act as noise which makes it tougher for the model to correctly learn how to model that particular dataset. Some examples of these characters are currency signs such as the pound sign, typographical symbols such as guitar or trumpet symbols and accented characters, which were replaced by the corresponding character without the accent.

Secondly, certain punctuation characters, such as "at" signs and the "greater-than"

sign were also removed. While it is possible that this later symbol was used in useful ways such as in math equations, it is used in Reddit as a formatting character, meaning that this character was predominately present in the dataset without giving it any useful additional information.

Thirdly, the models are trained using embeddings which are obtained at this stage and are related to the current dataset being used.

5.4 Evaluation Metrics

Two types of metrics were used to measure the performances of the models: the negative log-likelihood (or variational bound for the variational models) and the model's perplexity, which are intrinsic metrics, and embedding-based metrics, which are extrinsic metrics.

Both the negative log-likelihood and the perplexity measures how well the model is able to predict test data. The perplexity, ppl, is related to the negative log-likelihood, NLL, as shown in equation 5.1, where $\#tokens$ is the number of tokens on the test data.

$$ppl = \frac{e^{NLL}}{\#tokens} \tag{5.1}$$

In turn, for the variational models (VHRED and VHCR), the negative log-likelihood has the same intrinsic evaluation purpose as the sum of the reconstruction loss, explained in Section 2.3, and the KL divergence. The KL divergence, for this same evaluation, also allows us to measure the amount of information the model encoded in the latent variable, as seen in Section 3.2 and Section 3.3.

Currently, the most effective metrics to evaluate dialogue systems are embeddings-based metrics because instead of evaluating dialogue considering the words themselves, like the metrics described above, these embeddings-based metrics consider the meaning of said words by resorting to embeddings. Embeddings are vectors that represent the meaning of words and that approximate them based on their meaning; that is, if we have two different words with a similar meaning ("address" and "location", for example), their two embeddings are expected to be similar. These embeddings are generated by distributional semantics methods such as Word2Vec [39]. In general, these methods generate the embeddings by considering how often a word co-occurs with other words in the corpus. These embedding-based metrics usually approximate utterance-level embeddings by combining the vectors of all words in the utterance. A measure like the cosine similarity is then used to compare the textual similarity between the utterance-level embeddings of the candidate and of the target responses. For the experiments using the Ubuntu and askIT datasets, the Word2Vec embeddings trained on the Google News Corpus were used, following established practice [40, 41, 42]. For the experiments using the Portuguese dataset, portuguese embeddings were used [43].

The Average metric, which was proposed by Foltz et al. in 1998 [40], calculates utterance-level embeddings of the reference response and of the model's response by averaging the vector representations of their constituent words. Then, the cosine similarity is computed between the two utterance-level embeddings.

The Extrema metric, introduced by Forgues et al. in 2014 [41], is similar to the average metric, except that for each dimension of the word vectors, it takes the most extreme value among all word vectors in the utterance, instead of the mean, and uses this value in the utterance-level embedding. This approach prioritizes informative words over common ones; words that appear in similar contexts will be close together in the vector space. Thus, common words are pulled towards the origin because they occur in many different contexts, while words carrying important semantic information will lie further away. By taking the extrema along each dimension, we are thus more likely to ignore common words.

The Greedy matching metric was first introduced by Rus et al. in 2012 [42] and it does not compute utterance-level embeddings. This metric first finds the best non-exclusive word alignments between the model response and the reference response, and then computes the mean over the cosine similarity between the aligned words. As this metric is asymmetric, it then averages the greedy matching scores in each direction, i.e. from the model response to the reference response and from the reference response to the model response. This approach favors responses with key words that are semantically similar to those in the reference response.

5.5 Results and Discussion

For practical reasons, after testing how long an epoch took on the largest corpus, it was decided that for every model and every corpus, 30 epochs were going to be run for a total training time of approximately 68 hours for the Ubuntu corpus with the VHCR model.

For the Ubuntu dataset, on average, it takes about 2 hours and 20 minutes to train an epoch using the VHCR model, 2 hours and 15 minutes when using the VHRED model and about 2 hours when using HRED. For the askIT dataset, a considerably smaller dataset, it takes, on average, about 23 minutes to train an epoch using VHCR, 22 minutes when using VHRED and 16 minutes when using HRED.

Every VHCR trained model has a size of about 324 megabytes, while a VHRED model has a size of 240 megabytes and HRED has a size of 215 megabytes.

5.5.1 Evaluation with Negative Log-Likelihood

The results of both negative log-likelihood and perplexity of the models trained on Ubuntu are presented in Table 5.1. For VHRED and VHCR, the values of the measured KL divergence are also presented in the same table.

Model	Epoch	NLL	Word perplexity	KL divergence
HRED	1	4.077	58.968	-
	2	3.942	51.526	-
	5	**3.824**	**45.804**	-
	10	3.827	45.918	-
	15	3.865	47.711	-
	20	3.904	49.595	-
	25	3.934	51.126	-
	30	3.957	52.284	-
VHRED	1	**4.666**	**106.315**	1.002
	2	4.698	109.683	1.568
	5	4.846	127.218	2.399
	10	4.766	117.427	2.784
	15	5.515	248.484	4.046
	20	5.625	277.298	4.335
	25	5.211	183.309	3.981
	30	5.251	190.777	4.130
VHCR	1	5.054	156.672	1.555
	2	5.412	223.982	2.703
	5	4.155	63.731	1.074
	10	3.937	51.271	0.719
	15	3.881	48.473	0.669
	20	**3.865**	**47.724**	0.626
	25	3.913	50.034	0.784
	30	3.902	49.502	0.714

Table 5.1: Negative Log-Likelihood Evaluation Results of Models Trained on Ubuntu (lower scores represent better performance)

From these results, we can conclude that among all models, VHRED had the most difficulties in predicting the test set since it has the highest overall word perplexity value.

As previously mentioned, for this evaluation we can use the KL divergence to measure the amount of information encoded in the latent variable. As such, if the KL divergence is zero, we can affirm that the model completely ignored its latent variable (it degenerated). Despite having the highest overall word perplexity value, VHRED seem to have excessively resorted to the use of this latent variable, as the KL divergences measured are far higher than the ones measured on VHCR.

Among all trained models, HRED is able to achieve the lowest NLL after the fifth epoch. This was to be expected as this is the only model that is not a VAE model and therefore has a simpler approach when it comes to predicting its outputs.

The most interesting aspect of this evaluation was seeing that on the first few training epochs, VHCR was able to drastically improve its word perplexity scores. Being a VAE model, it can rival the HRED model's word perplexity scores meaning that while having

the capability of generating more diverse responses as explained in Section 2.3, this model is able to predict the test set in a more satisfactory way. It is to be noted that its KL divergence scores are lower than the ones of the VHRED model from the third epoch onwards. In fact, after the second epoch, the KL divergence gets considerably smaller and its perplexity scores greatly improve. This supports what was mentioned in VHCR's paper, a higher KL divergence does not necessarily lead to better performance and it is instead more important for models to balance the amount of information encoded in the latent variable and how closely these representations follow a Gaussian distribution.

Comparing these results to the ones measured on the original VHCR's paper [28], presented in Table 5.2, in which all three models were trained and evaluated on the Ubuntu dataset, all models except VHRED achieved comparable results. However, because of memory limitations, all models were here trained with a batch size of 30, instead of a batch size of 40 used by VHCR's authors, which may have contributed to a worse performance of the VHRED model.

Model	NLL	Word perplexity	KL divergence
HRED	3.766		-
VHRED	3.824	45.787	0.461
VHCR	3.951	51.987	0.756

Table 5.2: Original VHCR Paper's Negative Log-Likelihood Evaluation Results of Models Trained on Ubuntu (lower scores represent better performance)

The results obtained with the models trained on the askIT dataset, the perplexity, negative log-likelihood and, for VHRED and VHCR, the measured KL divergence are presented in Table 5.3.

These results are quite different than the ones obtained with Ubuntu. While HRED is the model that gets the lowest perplexity score here, VHRED is able to achieve a better score than VHCR. The variational models start with a word perplexity score much higher than HRED but are able to decrease it to comparable values at the end of the training. Being more complex, it is understandable that these models take more epochs to obtain their best performance. The word perplexities of VHRED and VHCR are here much more similar to each other than the ones obtained when training these models on the Ubuntu dataset and it is to be noted that the models used the latent variables in similar amounts.

The smaller size of the askIT dataset may explain the higher overall perplexity scores obtained with the models trained on this dataset than the scores obtained with the models trained on the Ubuntu dataset. Also, the fact that the average utterance length is considerably higher may contribute to the lower performance observed. In fact, as we will see below in Section 5.5.3, the models trained on this dataset tend to produce longer utterances than the ones trained over Ubuntu.

Finally, the results obtained with the models trained on the Portuguese dataset are

Model	Epoch	NLL	Word perplexity	KL divergence
HRED	1	4.729	113.226	-
	2	4.439	84.673	-
	5	4.173	64.934	-
	10	**4.140**	**62.799**	-
	15	4.275	71.846	-
	20	4.457	86.209	-
	25	4.636	103.121	-
	30	4.814	123.259	-
VHRED	1	5.107	165.150	0.439
	2	5.051	156.162	0.709
	5	5.615	274.391	1.885
	10	4.817	123.549	1.499
	15	4.550	94.671	1.377
	20	4.450	85.616	1.266
	25	**4.390**	**80.608**	1.147
	30	4.403	81.663	1.067
VHCR	1	5.534	253.078	0.905
	2	5.802	331.020	1.578
	5	5.504	245.656	1.828
	10	4.709	110.975	1.384
	15	4.613	100.830	1.386
	20	4.607	100.190	1.385
	25	4.600	99.450	1.306
	30	**4.594**	**98.918**	1.210

Table 5.3: Negative Log-Likelihood Evaluation Results of Models Trained on askIT (lower scores represent better performance)

presented in Table 5.4. As I was more interested in training a chatbot that could be used in a IT support scenario, this experiment was not performed to check which dataset among the three was able to get higher scores but instead, the goal here is to compare the used models when trained on a dataset that has a considerably larger domain and is written in a language other than English.

When comparing the model's word perplexity scores, HRED performed better than the variational models. Additionally, the values obtained by VHCR got progressively worse and never improved throughout the epochs. However, as we will see in the section below, this does not necessarily mean that the model did not improve throughout its training. Instead, it may mean that in order to be able to generalize to the training data, it had to learn how to adapt to this data in such a way that made the learning process possible which may have lead to a drift between the ground truth and the actual output utterances, which may still be correct. It is to be noted that the KL divergence got progressively higher, which as previously noted, seems to be related to an increase of the word per-

Model	Epoch	NLL	Word perplexity	KL divergence
HRED	1	4.925	137.714	-
	2	4.610	100.532	-
	5	4.096	60.078	-
	10	3.521	33.812	-
	15	3.118	22.606	-
	20	2.825	16.867	-
	25	2.611	13.609	-
	30	2.440	11.478	-
	35	**2.296**	**9.931**	-
VHRED	1	5.717	304.115	0.869
	2	5.832	341.140	1.407
	5	5.488	241.788	1.836
	10	4.690	108.823	1.655
	15	4.131	62.243	1.391
	20	**3.805**	**44.912**	1.209
	25	4.854	128.303	2.708
	30	5.003	148.831	3.088
	35	4.675	107.280	2.917
VHCR	1	**6.112**	**451.129**	1.367
	2	6.477	650.094	2.232
	5	6.578	718.855	3.205
	10	6.608	740.751	4.118
	15	6.909	1001.242	5.000
	20	7.453	1725.496	5.960
	25	7.743	2304.329	6.565
	30	8.186	3591.092	7.215
	35	8.259	3861.860	7.506

Table 5.4: Negative Log-Likelihood Evaluation Results of Models Trained on Portuguese (lower scores represent better performance)

plexity score. Because the best scores were mostly achieved at the latter epochs, for this dataset only, 35 epochs were run instead of the standard 30 that were established earlier.

5.5.2 Evaluation with Embedding-based Metrics

The results of the embedding-based evaluations of models trained on Ubuntu are presented on Table 5.5.

Let us start by analyzing the Embedding Average scores. The highest average score was obtained by VHCR and the second highest was obtained by HRED. These results are aligned with our expectations that the VHCR should be able to better encode the context of a conversation than its predecessors, HRED and VHRED.

Looking at the Vector Extrema scores, HRED was the best model between the three.

Model	Epoch	Average	Extrema	Greedy
HRED	1	**0.573**	0.336	**0.423**
	2	0.557	0.326	0.408
	5	0.570	**0.337**	0.415
	10	0.554	0.331	0.399
	15	0.548	0.326	0.393
	20	0.545	0.324	0.390
	25	0.541	0.320	0.387
	30	0.536	0.319	0.383
VHRED	1	**0.555**	**0.301**	**0.399**
	2	0.544	0.299	0.390
	5	0.542	0.300	0.387
	10	0.542	0.297	0.386
	15	0.539	0.292	0.382
	20	0.543	0.293	0.384
	25	0.538	0.290	0.379
	30	0.542	0.290	0.382
VHCR	1	0.577	0.310	0.425
	2	0.549	0.295	0.392
	5	**0.582**	0.314	0.424
	10	0.561	0.309	0.415
	15	0.560	0.312	0.421
	20	0.561	0.309	0.427
	25	0.555	0.309	0.425
	30	0.572	**0.316**	**0.436**

Table 5.5: Embedding-based Evaluation Results of Models Trained on Ubuntu (higher scores represent better performance)

This suggests that HRED is better able to write utterances with the more appropriate informative word.

Finally, we could see that in the Greedy Matching evaluation, VHCR outscored the other models. This suggests that, contrarily to what the extrema score hinted, the output utterances generated VHCR are not less informative than the ones obtained with HRED.

With these results in mind, we can conclude that when trained on the Ubuntu dataset, VHCR showed slightly overall better results on this embedding-based evaluation than the HRED model, and both of these performed better than VHRED.

One thing to note is that these metrics do not seem to improve with the extensive training of the HRED and VHRED models. This means that, at least for this kind of evaluation, there does not seem to exist any incentive to train these models for a large number of epochs as no improvement would apparently be made to the model.

Comparing these results to the measured ones on VHCR's paper, presented in Table 5.6, while there were some differences on the measurements, they were relatively similar.

Model	Average	Extrema	Greedy
HRED	0.567	0.337	0.412
VHRED	0.545	0.314	0.398
VHCR	0.570	0.312	0.425

Table 5.6: Original VHCR Paper's Embedding-based Evaluation Results of Models Trained on Ubuntu (higher scores represent better performance)

The results of the embeddings-based evaluations of models trained on the askIT dataset are presented on the Table 5.7. When trained on this dataset, we can see that all the models have higher scores compared to the ones obtained on the Ubuntu dataset. Here, the VHRED scores are very similar to the HRED ones and the VHCR, once again, performs better than the other two models.

Model	Epoch	Average	Extrema	Greedy
HRED	1	0.640	0.304	0.408
	2	0.650	0.321	0.420
	5	**0.662**	**0.349**	**0.443**
	10	0.616	0.340	0.403
	15	0.631	0.341	0.418
	20	0.633	0.344	0.42
	25	0.640	0.344	0.426
	30	0.636	0.342	0.424
VHRED	1	0.660	0.336	0.436
	2	**0.665**	0.336	**0.444**
	5	0.650	0.326	0.430
	10	0.654	0.333	0.434
	15	0.654	0.337	0.436
	20	0.656	0.337	0.438
	25	0.659	0.338	0.443
	30	0.656	**0.339**	0.439
VHCR	1	0.670	**0.358**	0.450
	2	0.639	0.310	0.407
	5	0.655	0.329	0.432
	10	0.662	0.339	0.442
	15	**0.687**	0.343	**0.466**
	20	0.682	0.336	0.462
	25	0.670	0.331	0.452
	30	0.677	0.338	0.461

Table 5.7: Embedding-based Evaluation Results of Models Trained on askIT (higher scores represent better performance)

In the Embedding Average metric, both VHRED and VHCR score better than HRED but their results are relatively similar.

In the Vector Extrema evaluation, VHCR is superior to the rest of the models. This suggests that this model is the model that can more easily output utterances in which more informative words are presented.

In the Greedy Matching evaluation, as with what happened with the Ubuntu dataset, VHCR is considerably better, meaning that VHCR is better able to output utterances that have key words semantically similar to the reference responses.

The results of the embeddings-based evaluations of models trained on Portuguese are presented on the Table 5.8. However, we must note that, as the dataset is composed of portuguese utterances, different embeddings were used and as such, these values can not be directly compared to the ones obtained with the other datasets (and embeddings).

Model	Epoch	Average	Extrema	Greedy
HRED	1	0.826	0.442	0.550
	2	0.779	0.418	0.520
	5	0.787	0.430	0.541
	10	0.822	0.459	0.583
	15	0.836	0.484	0.611
	20	0.838	0.497	0.623
	25	0.843	0.509	0.631
	30	0.852	0.521	0.646
	35	**0.852**	**0.525**	**0.652**
VHRED	1	0.793	**0.445**	0.543
	2	**0.831**	0.428	**0.569**
	5	0.827	0.419	0.563
	10	0.823	0.412	0.564
	15	0.812	0.407	0.559
	20	0.813	0.409	0.563
	25	0.812	0.407	0.562
	30	0.820	0.410	0.568
	35	0.820	0.410	0.568
VHCR	1	0.816	0.418	0.553
	2	0.826	0.425	0.562
	5	0.847	**0.425**	0.586
	10	0.846	0.415	0.580
	15	0.843	0.407	0.580
	20	0.846	0.412	0.583
	25	0.848	0.407	0.585
	30	**0.849**	0.409	**0.586**
	35	0.849	0.403	0.586

Table 5.8: Embedding-based Evaluation Results of Models Trained on Portuguese (higher scores represent better performance)

In the Embedding Average metric, the HRED model outscored both variational models and VHCR was better than VHRED.

In the Embedding Extrema metric, the HRED model outscored again both variational models, this time in a greater ratio. As for these latter models, while VHRED achieved the highest score on the first epoch, they achieved an overall similar performance throughout the rest of the epochs.

In the Embedding Greedy metric, the HRED model outscored yet again both variational models and VHCR outscored VHRED. It is to be noted that while the perplexity got higher scores, this evaluation assures us that the model did in fact perform better with its training.

This experiment allows us to conclude that HRED is in fact a state of the art model and that, while VHCR seems to be better than VHRED, the Variational Autoencoder models do not seem to be necessarily better than the other types of models.

5.5.3 Examples of Generated Responses

As a final evaluation, the implemented program presented in Section 3.4 was used to interact with the trained VHCR model and to see how the model could answer frequently asked questions that were not present in any of the used datasets used for training. For the models trained on the askIT and Ubuntu datasets, these questions were reformulated from ones found at `https://www.mcwareitsolutions.com/faq` for simplification purposes. Six questions from each of these distinct topics were used: Monitor FAQs; Windows FAQs; Network FAQs and VIRUS FAQs. For the model trained on the Portuguese datasets, some general questions about current events and entertainment and were used.

Some examples of the generated responses and the respective questions are presented in Tables 5.9 and 5.10. With the responses obtained, we can conclude that the model trained on Ubuntu outputs shorter utterances, which ultimately works better since the longer the output utterance, the harder it is for it to be close to an ideal response. As such, the models trained on Ubuntu seem to be better able to reply to the given questions. However, the models are still not capable of outputting satisfactory replies if our goal is to build a chatbot that is capable of answering these types of questions in a fully accurate way.

5.6 Results Overview

Overall, the chatbots trained on the Ubuntu dataset achieved better performance in the intrinsic evaluations. Interestingly, the extrinsic embeddings-based scores for the askIT dataset are higher, despite the generated responses appearing to be of lower quality (according to my own judgement) compared to the Ubuntu ones. As such, we can observe

Questions	askIT	Ubuntu
image on screen is distorted	maybe try a different version of windows if you can , and the same way as the os ? if not , then it s probably a good idea	black screen with a black screen ? if so , try to boot it up
how do I get into safe mode?	in theory , most modern tools will have to be a recovery disk . it will scan your computer and find out what s going on . most of the	assuming you 're using grub , press esc , and then choose the option to boot
how can I recover a file?	maybe a bit of googling , are you trying to copy the files to a file ? are you using a zip file ?	explain what happened ? what is the file name ? the file is corrupt ?
I can not connect to my network drive	assuming you are using the windows admin , you are using the same ip as the router . this is a common problem . i have a linksys ae	smbmount is the command line tool to connect to your server via ssh
can a virus damage computer hardware?	you could potentially dban the hard drive . the best way to do this is to install it and check for viruses .	possible to do it , but if you install windows , then install ubuntu
how do I delete internet cookies?	since the spying page is already the newest version , i would remove the java and then do that ? and would that be the same as well ?	firestarter is a frontend to iptables , which is what you need for your firewall

Table 5.9: Examples of the Generated Responses using the askIT and Ubuntu datasets

that the models trained on Ubuntu can produce more convincing text for the IT support domain.

Nevertheless, we must note that this is to be expected as the datasets that can be extracted with RDET, like the askIT dataset, either are of a small size or have entries from multiple subreddits, which relate to multiple, albeit possibly similar, fields.

The performance of the chatbots could perhaps be improved if such datasets were selected with shorter sentences. Reddit submissions may contain 40,000 characters and comments may contain 10,000 characters. As such, a dataset extracted from Reddit will most likely not be used on a dialogue system without possibly having its turns truncated, allowing them to only have a certain number of characters. This can be quite problematic as these characters that are kept may not be enough to represent the meaning of that particular turn.

One thing that was noticed at this stage was that the Reddit's tree-shaped nature dialogues introduces a new difficulty when training a model. Since each submission or

Questions	Portuguese
qual o melhor filme do ano?	talvez seja uma treta do facto de que os media de vida e bastante inteligente , mas a ideia da perda de treino . .
viste as noticias hoje?	assim por onde raio e que vao buscar o dinheiro a parte delas ? porque raio e que em portugal fica com algum tipo de duvida .
a seleccao joga amanha?	quando alguem tiver que ter de fazer algo de bom a partir de hoje em dia nao tenho a certeza absoluta o dia todo .
qual o valor das propinas?	e a de mais uma vez por semana de verao e com as financas . ainda me lembro de alguns meses para se baixar dos media

Table 5.10: Examples of the Generated Responses using the Portuguese datasets

comment may be replied giving rise to a new thread of conversation, many dialogues share the first turns, causing these utterances to be over-represented in the dataset. This causes models to update its hidden states more than once with the same utterances, which will give more importance to these ones. In one hand, this can be seen like a good thing since it can be argued that a comment that receives more replies is more interesting than one that did not get any, which is often the case since people can upvote the comments they think are better which in turn gives it more exposure and consequently more replies. However, inflamed comments can also provoke users to reply and if this difficulty was not treated, the model ultimately could mostly respond in an inflamed way (even though this kind of comments are not so common in most subreddits).

Chapter 6

Conclusion

This dissertation is concluded with this chapter in which I express some final notes that relates to the conclusions made possible by the development of this dissertation. Some ideas for future work are also presented. These could either potentially improve the performance of the chatbots or improve the quality of RDET.

6.1 Final Notes

In this dissertation, several contributions were made. First, an overview of the chatbot's field that includes its evaluation throughout the time and its current state is presented. Then, commonly used architectures and models are explained and Variational Autoencoder models are introduced. A new tool called RDET was developed which allows researchers to build new datasets using content from Reddit. Three state-of-the-art models (HRED, VHRED and VHCR) were reproduced using the Ubuntu Dialogue Corpus, as well as two newly generated datasets: askIT and Portuguese. Finally, these three models were evaluated using two different types of metrics: Negative Log-Likelihood and Embedding-based.

One the one hand, generative chatbots are still very much a recent technology and the results obtained can be disappointing because the system is not able to answer most questions asked in an acceptable way, sometimes even answering in unwanted ways such as saying "good luck" when asked a question, for instance. On the other hand, the idea that a model can learn to output understandable utterances by joining words one by one by itself is quite fascinating and improvements in the near future are to be expected since the major part of the techniques used in these kind of systems are very recent, most not being more than five years old.

Overall, it is easy to see why retrieval based chatbots seem to be the technology that provides better results for today's needs. While the models trained and evaluated in this thesis are not good enough to properly answer IT related questions in a real IT help desk scenario, the grammaticality and topic-appropriateness of generated answers are good

indicators that the technology is moving in the right direction. To provide insightful and useful answers in the IT domain, a model would have to truly understand the concepts related to this particular domain. Currently, these models generate utterances that while mostly grammatical correct, and at times convincing enough to people who may not be familiar with these concepts, do not make much sense in reality.

6.2 Future Work

In future work, the problem described in Section 5.6 should be addressed and it consists on the over-representation of some utterances on datasets extracted by RDET. To alleviate this problem, I propose that a model could check if it has already seen any given utterance as part of the training process and if so, the model should not update its states once again. Other alternative that could be used to alleviate this problem is to simply delete those repetitive sub-trees from the training set but by doing this, the context of new utterances would be deleted and thus these utterances would not properly be represented in the training dataset as they would be missing their context.

As RDET needs a list of subreddits as input from which it should extract dialogues, some mechanism could be added to automatically find interesting subreddits based on specified keywords. This would allow the user not to worry about which specific subreddits the tool should consider but instead just focus on the keyword the dataset should be related to.

Other improvement that could be made relates to a further treatment of the sentences that are contained in the dialogues. The datasets extracted possibly contain curse words. As this is something that in most cases should be avoided, a mechanism to identify and delete such words could be implemented.

A language identifier could also be useful as it would detect the dataset language and delete comments or dialogues written in other language. This would be particularly useful for the extracted Portuguese dataset.

It would also be interesting to explore in-depth how the model would behave when trained on a domain that is more subjective than IT. For instance, having a music or movies discussion dataset would possibly make a more interesting chatbot as these kind of discussions stem from personal opinions which can be argued to be easier to model than to model a strong understanding of technical concepts that may not be explicitly explained in the training dialogues.

Other than this, generational chatbots could be trained with datasets like Cornell Movie-Dialogue Corpus (presented in Section 4.3) to obtain personality based chatbots. It would be interesting to see multiple instances of the same model trained on datasets corresponding to different and unique characters. Taking advantage of RDET, it would even be possible to train multiple instances of the same model on dialogues extracted

from different subreddits to see how different would the chatbots respond to the same questions. This could lead to interesting results as it could lead to a study related to the different subreddit's general opinions about a given topic.

Glossary

AI	Artificial Intelligence.
DSTC	Dialog System Technology Challenge.
GAN	Generative Adversarial Network.
GRU	Gated Recurrent Units Network.
HRED	Hierarchical Recurrent Encoder-Decoder.
IT	Information Technology.
KL	Kullback-Leibler.
LSTM	Long Short-Term Memory Network.
MLP	Multilayer Perceptron.
NLL	Negative Log-Likelihood.
NLP	Natural Language Processing.
RDET	Reddit Dataset Extraction Tool.
RNN	Recurrent Neural Network.
Seq2Seq	Sequence to Sequence.
VAE	Variational Autoencoders.
VHCR	Variational Hierarchical Conversation RNNs.
VHRED	Variational Hierarchical Recurrent Encoder-Decoder.

Bibliography

[1] Alan Turing. I.—Computing Machinery and Intelligence. *Mind*, LIX(236):433–460, 1950.

[2] Joseph Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 1966.

[3] Kenneth Mark Colby. Ten criticisms of parry. *SIGART Bull.*, 1974.

[4] Rollo Carpenter. Jabberwacky. `http://www.jabberwacky.com/`, 1981. Accessed: 2019-05-27.

[5] Creative Labs. Dr. sbaitso. `https://en.wikipedia.org/wiki/Dr._Sbaitso`, 1991. Accessed: 2019-05-27.

[6] Richard S. Wallace. *The Anatomy of A.L.I.C.E.*, pages 181–210. 2009.

[7] Inc. ActiveBuddy. Smarterchild. `https://mashable.com/2012/10/25/instant-messaging-history/?europe=true`.

[8] David A. Ferrucci. Ibm's watson/deepqa. *SIGARCH Comput. Archit. News*, 2011.

[9] Erica Sadun and Steve Sande. *Talking to Siri: Learning the Language of Apple's Intelligent Assistant*. 2013.

[10] Huan Feng, Kassem Fawaz, and Kang G. Shin. Continuous authentication for voice assistants. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 343–355, 2017.

[11] Amanda Purington, Jessie G. Taft, Shruti Sannon, Natalya N. Bazarova, and Samuel Hardman Taylor. "alexa is my new bff": Social roles, user satisfaction, and personification of the amazon echo. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2853–2859, 2017.

[12] Microsoft. Tay. `https://www.techrepublic.com/article/why-microsofts-tay-ai-bot-went-wrong/`, 2016. Accessed: 2019-06-23.

[13] Mindbowser. Global chatbot trends report – 2017. `https://mindbowser.com/chatbot-market-survey-2017/`, 2017. Accessed: 2019-05-27.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, pages 1735–1780, 1997.

[15] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[16] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 1997.

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, 2014.

[18] Janarthanan Rajendran, Jatin Ganhotra, Satinder Singh, and Lazaros Polymenakos. Learning End-to-End Goal-Oriented Dialog with Multiple Answers. 2018.

[19] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. 2015.

[20] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning End-to-End Goal-Oriented Dialog. 2016.

[21] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-To-End Memory Networks. 2015.

[22] Dana H. Ballard. Modular learning in neural networks. In *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1*, AAAI'87, pages 279–284, 1987.

[23] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. 2013.

[24] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1278–1286, 2014.

[25] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3783, 2016.

[26] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3295–3301, 2017.

[27] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

[28] Yookoon Park, Jaemin Cho, and Gunhee Kim. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1792–1801, 2018.

[29] R. Lowe, N. Pow, I. V. Serban, and J. Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. 2016.

[30] Jason D. Williams, Antoine Raux, and Matthew Henderson. The dialog state tracking challenge series: A review.

[31] Seokhwan Kim, Luis D'Haro, Rafael Banchs, Jason Williams, and Matthew Henderson. The fourth dialog state tracking challenge. 2016.

[32] S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, M. Henderson, and K. Yoshino. The fifth dialog state tracking challenge. 2016.

[33] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[34] Amazon. Alexa. `https://www.alexa.com`, 2019. Accessed: 2019-06-23.

[35] Jason Baumgartner. Pushshift. `https://pushshift.io/`. Accessed: 2019-05-25.

[36] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, 2002.

[37] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[38] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. 2004.

[39] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, 2013.

[40] P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25:285–307, 1998.

[41] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2, 2014.

[42] Vasile Rus and Mihai Lintean. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, NAACL HLT '12, pages 157–162, 2012.

[43] João António Rodrigues, António Branco, Steven Neale, and João Ricardo Silva. Lx-dsemvectors: Distributional semantics models for portuguese. In *PROPOR*, 2016.